1198-3878
US

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: １９９９年 ２月１９日

出 願 番 号
Application Number: 平成１１年特許願第０４１０３２号

出 願 人
Applicant (s): 株式会社日立製作所

CERTIFIED COPY OF
PRIORITY DOCUMENT

１９９９年１１月 ５日

特許庁長官
Commissioner,
Patent Office 近藤隆

【書類名】　特許願

【整理番号】　1198038781

【あて先】　特許庁長官　殿

【国際特許分類】　G06F　9/46

【発明の名称】　複数のオペレーティングシステムを実行する計算機

【請求項の数】　10

【発明者】

【住所又は居所】　茨城県日立市大みか町七丁目１番１号
株式会社　日立製作所　日立研究所内

【氏名】　齊藤　雅彦

【発明者】

【住所又は居所】　茨城県日立市大みか町七丁目１番１号
株式会社　日立製作所　日立研究所内

【氏名】　上脇　　正

【発明者】

【住所又は居所】　茨城県日立市大みか町五丁目２番１号
株式会社　日立製作所　大みか工場内

【氏名】　中村　智明

【発明者】

【住所又は居所】　茨城県日立市大みか町五丁目２番１号
株式会社　日立製作所　大みか工場内

【氏名】　大野　　洋

【発明者】

【住所又は居所】　神奈川県川崎市麻生区王禅寺１０９９番地
株式会社　日立製作所　システム開発研究所内

【氏名】　井上　太郎

【特許出願人】

【識別番号】　000005108

【氏名又は名称】　株式会社　日立製作所

【代理人】

【識別番号】　100068504

【弁理士】

【氏名又は名称】　小川　勝男

【電話番号】　03-3212-1111

【手数料の表示】

【予納台帳番号】　013088

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　複数のオペレーティングシステムを実行する計算機

【特許請求の範囲】

【請求項１】

複数のオペレーティングシステムと、それぞれの上記オペレーティングシステムのプログラム上で実行される複数のプロセスまたはスレッドとを記憶したメモリと、

それぞれの上記プロセス又はスレッドに割り当てられた優先順位に基づいて、オペレーティングシステムを実行する処理装置とを有する計算機であって、

上記処理装置は、それぞれの上記オペレーティングシステムで実行すべきプロセス又はスレッドの優先順位を読み出し、それぞれ読み出された優先順位を複数の上記オペレーティングシステム間で共通の優先順位に変換し、上記変換結果に基づいて優先して実行すべきオペレーティングシステムを決定するとともに、上記決定されたオペレーティングシステムを実行する計算機。

【請求項２】

請求項１において、上記メモリは複数の上記オペレーティングシステムで実行すべきプロセス又はスレッドの優先順位を上記共通の優先順位にマッピングするための優先順位変換テーブルを有し、上記プロセッサは上記変換テーブルに基づいて優先して実行すべきオペレーティングシステムを決定する計算機。

【請求項３】

請求項１又は２において、上記プロセッサは、複数のオペレーティングシステム間で共通の優先順位から、それぞれの上記オペレーティングシステムにおける優先順位を決定し、それぞれのオペレーティングシステムで実行される複数の上記プロセス又はスレッドの優先順位を変更する計算機。

【請求項４】

請求項３において、上記メモリは、上記共通の優先順位をそれぞれの上記オペレーティングシステムにおける優先順位にマッピングするための優先順位逆変換テーブルを有し、上記プロセッサは、上記優先順位逆変換テーブルに基づいて複数の上記プロセス又はスレッドの優先順位を変更する計算機。

【請求項５】

請求項１乃至４において、上記プロセッサはプロセス又はスレッドが指定された処理を実行する場合に、該プロセス又はスレッドを実行するオペレーティングシステムの優先順位を上昇させ、上記プロセス又はスレッドが該指定された処理を終了した場合に、上記オペレーティングシステムの優先順位を下降させる計算機。

【請求項６】

複数のプロセスまたはスレッドと、

複数の上記プロセス又はスレッドを実行すると共に、実行しているプロセス又はスレッドの優先順位を通知する複数のオペレーティングシステムと、

それぞれの上記オペレーティングシステムから通知された優先順位をオペレーティングシステム間で共通の優先順位に変換する優先順位変換処理と、

上記優先順位変換処理によって得られた共通の優先順位を比較して、より高い共通の優先順位を有するオペレーティングシステムを優先的に実行させる優先順位比較処理とを記憶した情報記憶媒体。

【請求項７】

複数のオペレーティングシステムを選択的に実行するオペレーティングシステム実行方法であって、

優先順位変換処理によって、それぞれの上記オペレーティングシステムで実行されるプロセス又はスレッドの優先順位をオペレーティングシステム間で共通の優先順位に変換し、

優先順位比較処理によって、上記優先順位変換処理によって得られた共通の優先順位を比較して、より高い共通の優先順位を有するオペレーティングシステムを優先的に実行させるオペレーティングシステム実行方法。

【請求項８】

請求項７において、上記優先順位変換処理はそれぞれのオペレーティングシステムの優先順位を上記共通の優先順位に変換するにあたって、異なったオペレーティングシステムの優先順位は互いに異なった共通の優先順位に変換するオペレーティングシステム実行方法。

【請求項９】

請求項７又は８において、前記優先順位変換処理は、それぞれの上記オペレーティングシステムが実行しているプロセスまたはスレッドの優先順位を上記共通の優先順位に変換するほかに、少なくとも、割込み処理中の状態，オペレーティングシステム自体の機能を実行中の状態，実行すべき処理が存在しない状態という三つの状態を共通の優先順位に変換するオペレーティングシステム実行方法。

【請求項１０】

複数のプロセスまたはスレッドを該複数のプロセスまたはスレッドに割り当てられた優先順位にしたがって実行する複数個のオペレーティングシステムと、該複数個のオペレーティングシステムを切替える手段とを有する計算機システムにおいて、該複数個のオペレーティングシステムは、それぞれが実行しているプロセスまたはスレッドの優先順位を計算機システム内で共通の優先順位に変換する優先順位変換手段と、該優先順位変換手段から得られた共通の優先順位を該オペレーティングシステムに切替える手段に通知する優先順位通知手段とを有し、該オペレーティングシステムを切替える手段は、それぞれのオペレーティングシステムから通知された共通の優先順位を比較して、より高い共通の優先順位を有するオペレーティングシステムを優先的に実行させる優先順位比較手段を有したことを特徴とする計算機システム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は複数個のオペレーティングシステムを切り替えつつ動作させる計算機およびそのオペレーティングシステム切り替え方式を対象分野とし、特に、相異なる優先順位体系を有するオペレーティングシステムの切替え方式に関連する。

【０００２】

【従来の技術】

一つの計算機で複数のオペレーティングシステムを動作させる技術として、従来から大型計算機において、「仮想計算機(Virtual Machine：ＶＭ）」が知られている。該従来技術では、計算機内で並行して動作する複数個の仮想計算機それ

ぞれの上で複数のユーザタスク（以降、プロセスおよびスレッドを統一してタスクと呼ぶ）を切り替えて処理を実行する。仮想計算機は、通常、大型計算機内の一つのプロセスとして実現されるが、仮想計算機とユーザタスクとの関連を考慮すると、一つのオペレーティングシステムであると見なすことも可能である。

【０００３】

一般に、それぞれの仮想計算機には、該仮想計算機の優先順位にしたがって一定のタイムスライス（ＣＰＵ割当て時間）を与える。各仮想計算機は、与えられたタイムスライスの範囲内でユーザタスクを切り替えて動作させる。このような仮想計算機技術の実行効率を高める技術として、特開平5－197577 号に開示された「仮想計算機における仮想計算機実行プライオリティ制御方式」がある。

【０００４】

該従来技術は、複数の仮想計算機と、該複数の仮想計算機を制御するための仮想計算機モニタとから構成される。仮想計算機は、システムタスクのように実行優先順位の高いタスクの実行開始・実行終了時に、仮想計算機モニタに対してタスクの優先順位を通知する。これを受けた仮想計算機モニタは、仮想計算機の実行優先順位を、通知された優先順位に変更する。仮想計算機の実行優先順位を仮想計算機内で実行されるタスクの優先順位に変更することにより、仮想計算機制御を効率的に行うことができるとしている。

【０００５】

【発明が解決しようとする課題】

マイクロプロセッサ、特に、組込み向けマイクロプロセッサの性能向上と、オペレーティングシステムの機能向上に伴って、複数個の相違なるオペレーティングシステムを一つの計算機上で同時に動作させ、これらを動的に切り替えて処理を行うというユーザニーズが現われている。

【０００６】

一般に、工場・プラントなどの機械制御，車載ナビゲーションシステムといった分野では、外部環境変化に対して即座に応答するというリアルタイム性，長時間連続運転を可能とする信頼性などが重要視されている。このため、割込み応答性が高く、かつ、コンパクトでモジュール構成を有するリアルタイム用オペレー

ティングシステム（リアルタイムＯＳ）が用いられることが多い。しかしながら、リアルタイムＯＳは、リアルタイム性・信頼性を重要視する反面、人間とのインタフェースに優れているとは言い難い。

【０００７】

一方、一般のパソコン（ＰＣ）に用いられている事務処理用オペレーティングシステム（事務処理ＯＳ）は、画像を用いて操作を行えるようにするなど、対人間インタフェースに優れている。このため、従来はリアルタイムＯＳを使用していた分野にも事務処理ＯＳのユーザインタフェースを使用したいという要求が高まっている。しかし、事務処理ＯＳは人間との対話を主な処理としているため、割込み応答性よりも処理のスループットを重要視し、比較的長時間割込み禁止状態のまま処理を実行することもある。また、コンパクトな構成を有するリアルタイムＯＳに対し、信頼性において匹敵するとは言い難く、２４時間連続運転などに向いていないという側面もある。

【０００８】

しかしながら、大型計算機上で複数の仮想計算機(オペレーティングシステム)を並行して動作させる方式と同様に、同一計算機上で事務処理ＯＳとリアルタイムＯＳとを動作させ、必要に応じてこれらのオペレーティングシステムを切り替えることができれば、優れたユーザインタフェースとリアルタイム性・信頼性とを両立させることができる。マイクロプロセッサの性能向上を考慮すれば、複数のオペレーティングシステムを一つの計算機上で動作させることは、大型計算機だけに許された技術ではなくなってきている。

【０００９】

このとき、それぞれのオペレーティングシステムの重要性を考慮すると、常にリアルタイムＯＳを優先的に動作させ、リアルタイムＯＳ上で実行させるタスクが存在しなくなった場合にのみ、事務処理ＯＳを動作させるという方法が最も単純なオペレーティングシステム切替え方式である。しかしながら、各オペレーティングシステム上で動作する個々のタスクの重要性が、このように（リアルタイムＯＳ上のタスクは、常に、事務処理ＯＳ上のタスクよりも重要性が高いというように）単純に切り分けられるとは限らない。

【００１０】

図２７に単純な切り分けを行えない例を示す。図２７は車載ナビゲーションシステムの構成例を示したものである。ここでは、システムを簡略化し、車載ナビゲーションシステムが、（１）運転位置を認識する位置認識タスク３７０，（２）目標地点までの最短経路を導出する経路探索タスク３７１，（３）ボタン・タッチパネルなどからの入力を処理するインタフェースタスク３７２，（４）運転休憩中に起動していたゲームタスク３７３、という四つのタスクから構成されていると仮定する。通常、高速応答性・信頼性が求められる位置認識タスク・経路探索タスクはリアルタイムＯＳ１１１上で実行され、ユーザインタフェースに優れた事務処理ＯＳ１１０では、インタフェースタスク・ゲームタスクなどを実行する。しかしながら、一般に、経路探索は非常に計算量の大きい処理であり、数秒オーダの計算時間を要することもある。上記に示した単純な重要性の切り分けを行うと、この間、インタフェースタスクの処理は停止し、ユーザがいくらボタンを押しても認識されないという問題が発生する。

【００１１】

図２７に示した車載ナビゲーションシステムではインタフェースタスクの重要性が高く、これが実行可能状態に移行した場合には、事務処理ＯＳを優先的に実行する必要がある。しかしながら、前記従来技術「仮想計算機における仮想計算機実行プライオリティ制御方式」は、各仮想計算機(オペレーティングシステム)の機能が同一であることを前提としている。一般に、リアルタイムＯＳは環境変化に高速に応答するため、事務処理ＯＳに比べて極めて多くの優先順位レベルを有することが多い。また、極端な場合、リアルタイムＯＳ上では、優先順位の「数値」が小さい（０に近い）ほど優先順位が高く、事務処理ＯＳ上では、優先順位の「数値」が大きいほど優先順位が高い、といった逆の意味付けがなされていることもある。このような場合、たとえ二つのオペレーティングシステムから優先順位を通知されても、いずれのオペレーティングシステムを優先的に起動してよいか判断できないことになる。このため、前記従来技術では、異なった機能を有する事務処理ＯＳとリアルタイムＯＳとを統一して管理することはできない。

【００１２】

本発明の目的は、複数の相異なるオペレーティングシステムを一つの計算機上で動作させるマルチオペレーティングシステム制御装置において、各オペレーティングシステム上で実行されているタスクの重要性を考慮して、動作させるべきオペレーティングシステムを選択・切り替えることであり、これによって、重要性のあるタスクが優先的に実行されるようにすることである。

【００１３】

【課題を解決するための手段】

上記目的を達成するため、本発明では、複数のタスクをそれぞれのタスクに割当てられた優先順位にしたがって実行する複数のオペレーティングシステムと、この複数のオペレーティングシステムを切り替えて実行する計算機において、下記に示す処理を行う。

【００１４】

（１）各オペレーティングシステムが実行しているタスクの優先順位を確認する優先順位監視処理を行う。若しくは、各オペレーティングシステム内に、自分が実行しているタスクの優先順位を通知する優先順位通知処理を行う。事務処理ＯＳ，リアルタイムＯＳでは、内部に優先順位通知処理を追加することが不可能なこともある。このとき、前者の優先順位監視処理行う必要がある。

【００１５】

（２）各オペレーティングシステムから入手した実行中タスクの優先順位を、それぞれのオペレーティングシステムで共通の優先順位に変換する優先順位変換処理を行う(以降、該共通の優先順位のことを「正規化優先順位」と呼ぶ。)。

（３）優先順位変換処理によって得られた各オペレーティングシステムの正規化優先順位を比較して、動作させるべきオペレーティングシステムを決定し、かつ、オペレーティングシステムの切り替えを実行する優先順位比較処理を行う。

【００１６】

本発明によれば、優先順位通知処理は各オペレーティングシステム内に設けられ、オペレーティングシステムがタスク切り替えを行うたびに、該タスクの優先

順位を通知する。

【００１７】

タスク切り替え機能はオペレーティングシステムの中核を構成するものであるため、市販されているオペレーティングシステム内に優先順位通知手段を組込むことが不可能なことも多い。しかしながら、一般に、オペレーティングシステムは実行中タスクの管理情報を計算機のメモリ上に保持し、この管理情報の一部にタスクの実行優先順位がある。また、タスク切り替え処理を高速化するため、現在実行中のタスクの優先順位を特定の変数（優先順位保持変数）に格納しておく場合もある。優先順位監視処理は、タスク管理情報または優先順位保持変数を読み出して、各オペレーティングシステムが実行しているタスクの優先順位を確認する。優先順位監視処理は、外部割込み，タイマ割込みなどのタイミングで優先順位の確認を行う。

【００１８】

優先順位変換処理は、各オペレーティングシステムから入手した実行中タスクの優先順位を、計算機内で共通の正規化優先順位に変換する。このため、優先順位変換処理は、各オペレーティングシステムに対応する優先順位変換テーブルを有することがある。優先順位変換テーブルは、各オペレーティングシステム固有の優先順位から正規化優先順位を取り出すための対応表である。簡単な数式によって個々のオペレーティングシステムの優先順位を正規化優先順位に変換することも可能であるが、高速かつ柔軟なオペレーティングシステム切り替えを実行するためには、優先順位変換テーブルを用いるほうがよい。例えば、優先順位変換テーブルに格納する正規化優先順位の値を適切に定めることにより、個々のオペレーティングシステムから得られる正規化優先順位が互いに等しくならないように変換を行うことも可能である。

【００１９】

優先順位比較処理は、各オペレーティングシステムの正規化優先順位を優先順位変換処理から入手し、現在動作させているオペレーティングシステムより高い正規化優先順位を有するオペレーティングシステムが存在すれば、オペレーティングシステムの切り替えを行う。

【００２０】

【発明の実施の形態】

本発明の実施例を図面を用いて説明する。

【００２１】

図１に本発明の第一の実施例における全体構成を示す。通常、計算機は、プロセッサ１００，メモリ１０１，入出力制御装置１０２，ディスク装置１０４，ディスプレイ１０５などから構成される。プロセッサ１００，メモリ１０１，入出力制御装置１０２はプロセッサバス１０３によって接続される。プロセッサ100は複数個のオペレーティングシステムを動作させるためのマイクロプロセッサである。メモリ１０１は、オペレーティングシステムである事務処理ＯＳ１１０，リアルタイムＯＳ１１１，各オペレーティングシステム上で動作するタスク112～１１７，オペレーティングシステム切り替えプログラム１１８を記憶する。これらのプログラムはプロセッサ１００によって読み出されて実行される。

【００２２】

入出力制御装置１０２には、プログラム・データを保存するためのディスク装置１０４，画面表示用デバイスであるディスプレイ１０５が接続される。また、工場・プラント制御用、あるいは、組込み向け計算機を実現する場合、入出力制御装置１０２には、リアルタイム制御ネットワーク１０６が接続されることがある。リアルタイム制御ネットワーク１０６には、センサ，アクチュエータなどの入出力機器が接続される。なお、入出力制御装置１０２に接続されるディスク装置１０４，ディスプレイ１０５，リアルタイム制御ネットワーク１０６の入出力装置１０４～１０６のうち、いずれかまたは全ては、システム構成によって省略されることもある。入出力制御装置１０２は、割込み信号線１０７によってプロセッサ１００と接続され、入出力動作完了などを通知することができる。図１中では、説明のため、割込み信号線１０７とプロセッサバス１０３が別の装置であるように記載しているが、実際には、割込み信号線１０７はプロセッサバス103の一部である。プロセッサ１００内部にはタイマ装置１０８が設けられており、一定周期毎に内部割込みを発生させる。タイマ装置１０８からの割込みは、オペレーティングシステムの計時などに使用される。

【００２３】

プロセッサ１００は割込み信号線１０７によって通知される外部割込みやタイマ装置１０８などからの内部割込みをマスクできる機能を装備する。割込みマスクとは、プログラムが割込みマスクを解除するまで、特定の割込みが入ることを遅延させる機能である。一般に、割込みマスク機能には、下記の三種類が存在する。

【００２４】

（１）全割込みマスク：全ての割込みをマスクする。

【００２５】

（２）個別割込みマスク：個々の割込みをそれぞれマスクできるようにする。

【００２６】

（３）割込みレベルマスク：各割込みにレベルを設定し、指定レベル以下の割込みをマスクする。

【００２７】

プロセッサ１００の種類によって、上記（１）（２）の組合せ、または、（１）（３）の組合せのいずれかで構成することが多い。後者の組合せで構成するプロセッサを採用する場合、対応する入出力装置の重要性にしたがって割込みレベルを割当てることになる。例えば、リアルタイム制御ネットワーク１０６からの割込みを、ディスク装置１０４，ディスプレイ１０５などからの割込みより高いレベルに設定する。

【００２８】

本実施例では、計算機内に二つのオペレーティングシステムである事務処理ＯＳ１１０，リアルタイムＯＳ１１１が存在する。このオペレーティングシステムは、各自に割当てられたメモリ・プロセッサ資源を用い、それぞれ、タスク１１２～１１４，タスク１１５～１１７を実行させる。ここでは、オペレーティングシステム数が２個、タスク数が６個（各オペレーティングシステムに対して３個ずつ）の例を示しているが、これらの数値よりも多い、または、少ないオペレーティングシステム，タスクを実装することも可能である。本実施例では、オペレーティングシステム数の動的な変更は想定しないが、各オペレーティングシ

ステムが動的にタスク生成・削除を行うことは可能である。また、本実施例では、事務処理ＯＳ１１０，リアルタイムＯＳ１１１について説明するが、実際に、各オペレーティングシステムがいかなる種類であっても、本発明で述べる技術は適用可能である。タスク１１２～１１４は事務処理ＯＳで実行される事務処理タスクであり、タスク１１５～１１７はリアルタイムＯＳで実行されるリアルタイムタスクである。事務処理ＯＳ１１０とリアルタイムＯＳ１１１は、各々独立にタスクの優先順位を定義している。図１に示す実施例では、事務処理ＯＳ１１０環境下で、タスク１１２～１１４が、それぞれ、優先順位０，７，３１を有し、リアルタイムＯＳ１１１環境下で、タスク１１５～１１７が、それぞれ、優先順位０，９９，２５５を有する。尚、本実施例においてタスクとはプロセス又はスレッドの総称を意味する。プロセスとは、個別のメモリ空間を有して動作するプログラムを意味する。このため、通常あるプロセスは別のプロセスのデータを変更することはできない。スレッドとは、プロセスのメモリ空間を共有して動作するプログラムを指す。従って、同一プロセス内で動作するスレッド同士の間ではデータ保護機能は存在しない。

【００２９】

各オペレーティングシステム内には、優先順位通知モジュール１２０，１２１が存在する。優先順位通知モジュール１２０，１２１は、各オペレーティングシステムのタスク切り替え時に実行され、オペレーティングシステム切り替えプログラム１１８に対して次に実行すべきタスクの優先順位を通知する。

【００３０】

一般に、事務処理ＯＳとリアルタイムＯＳとは異なった優先順位体系を有することが多い。リアルタイムＯＳでは、比較的多くの優先順位レベルを有することにより、重要な割込みに対する高速応答性を実現する。一方、事務処理ＯＳでは、比較的少ない優先順位レベルを用いて、スループットを向上させる方式を採用する。また、オペレーティングシステムの種類によっては、優先順位の数値が小さい（０に近い）ほど優先順位が高い場合と、優先順位の数値が大きいほど優先順位が高い場合とがある。このため、優先順位通知モジュール１２０，１２１から得られた優先順位をそのまま比較しても意味がない。なお、以降、事務処理

ＯＳ１１０では、優先順位の数値が大きいほどタスクの優先順位が高く（図１中では、タスク１１４が最も優先される）、リアルタイムＯＳ１１１では、逆に、優先順位の数値が小さいほどタスクの優先順位が高い（図１中では、タスク115が最も優先される）ものとして説明する。また、事務処理ＯＳ１１０における優先順位は０～３１の範囲で指定でき、リアルタイムＯＳ１１１における優先順位は０～２５５の範囲で指定できるものとする。

【００３１】

二つのオペレーティングシステムにおける優先順位体系の違いを解消するため、オペレーティングシステム切り替えプログラム１１８は、優先順位変換モジュール１２２，１２３と優先順位比較モジュール１２４とを構成要素として有する。優先順位変換モジュール１２２，１２３は、それぞれ、事務処理ＯＳ１１０，リアルタイムＯＳ１１１から得られた優先順位をシステム内で共通の比較基準である正規化優先順位に変換する。各オペレーティングシステムの優先順位をそのまま比較することは不可能であるが、一旦両者を正規化優先順位に変換すれば、両オペレーティングシステムが実行しているタスクの優先順位を比較することができる。なお、ここで、正規化優先順位が一方のオペレーティングシステムの優先順位と全く同一であってもよい。この場合、当該オペレーティングシステムの優先順位がそのまま正規化優先順位となる。優先順位比較モジュール１２４は、両オペレーティングシステムから得られた正規化優先順位を比較し、かつ、より高い正規化優先順位を有するオペレーティングシステムに切り替えて実行させる役割を有する。

【００３２】

図２に、プロセッサ１００の内部構成例を示す。キャッシュメモリ１３１はメモリ１０１上のデータまたは命令を一時的に格納するバッファ記憶装置である。CPU130は演算回路であり、メモリ１０１若しくはキャッシュメモリ１３１上に存在する命令を順次読み出して実行する。命令実行に際して、演算結果を一時的に保持するための汎用レジスタ１３２，命令アドレスを指定するプログラムカウンタ１３３，実行状態を保持するステータスレジスタ１３４を用いる。CPU130，キャッシュメモリ１３１，汎用レジスタ１３２，プログラムカウンタ１３３，ステ

ータスレジスタ１３４は、互いに、データ転送を行う複数の信号線であるデータバス１３６，アドレス指定を行う複数の信号線であるアドレスバス１３７によって接続されている。

【００３３】

割込み信号線１０７とタイマ装置１０８は割込みコントローラ１３５に接続される。割込みコントローラ１３５は、CPU130に対して割込み状態信号１３８を生成する役割を有する。割込み状態信号１３８は、プロセッサ１００に対して現在如何なる種類の割込みが発生しているかを示す信号である。通常、ステータスレジスタ１３４は現在の割込みマスクに関する情報を有しており、割込み状態信号１３８で指定される割込みを受け付けるか否かを決定する。割込みを受け付ける場合、割込みコントローラ１３５は、プログラムカウンタ１３３，ステータスレジスタ１３４などの値を書き替え、対応する割込み処理プログラムを実行させる。

【００３４】

ステータスレジスタ１３４の構成を図３に示す。ここでは、プロセッサ１００が全割込みマスク機能と割込みレベルマスク機能とを装備している場合を示している。図３では、ステータスレジスタ１３４が割込みブロックビット１４０と割込みマスクレベルフィールド１４１を有する。割込みブロックビット１４０がＯＮである場合、プロセッサ１００に対する全ての割込みがマスクされる。割込みマスクレベルフィールド１４１は現在の割込みマスクレベル値を示し、これ以下の割込みレベルは受け付けられない。図３では、割込みマスクレベルフィールド１４１は４ビット長である。このため、合計１６種類のマスクレベルを指定可能であるが（通常、割込みレベル０は「割込みが発生していない」ことを意味し、割込みマスクレベルを０とすることは「割込みマスクを行わない」ことを意味するため、実質的には１５種類となる）、割込みマスクレベルフィールド１４１のビット数を変更することにより、受理できる割込みレベルの種類を増減させることができる。

【００３５】

図４は、プロセッサ１００が全割込みマスク機能と個別割込みマスク機能とを

装備している場合のステータスレジスタ１３４の構成を示したものである。この例では、ステータスレジスタ１３４は実際には二つのレジスタ（実行状態レジスタ１４２と割込みマスクレジスタ１４３）から構成される。図３と同様、実行状態レジスタ１４２内に割込みブロックビット１４０が設けられている。割込みマスクレジスタ１４３内の割込みマスクビット１４４～１４７はそれぞれ別々の割込みに対応しており、この割込みマスクビットのいずれかをＯＮとした場合、対応する割込みが受け付けられなくなる。図３に示すステータスレジスタは、図４のステータスレジスタの特殊なものである。例えば、割込みマスクビット１４４のみＯＮとなっている状態をレベル１とし、割込みマスクビット１４４，１４５の二つがＯＮとなっている状態をレベル２、割込みマスクビット１４４～１４６の三つがＯＮとなっている状態をレベル３，…、のように対応させることができる。このため、以降、ステータスレジスタ１３４は図４に示す構成として、本発明の説明を行う。

【００３６】

一般的なプロセッサでは、割込みを受理すると、ハードウェアによって自動的に割込みブロックビット１４０がＯＮに書き替えられ、プログラムカウンタ133に割込み処理プログラムのアドレスが格納される。必要に応じて、割込み処理プログラムが割込みブロックビット１４０をＯＦＦに書き替えて割込み受け付けを許可することができる。また、オペレーティングシステムおよびタスクが、一時的に割込みブロックビット１４０や割込みマスクレジスタ１４３の内容を書き替え、特定割込みの受け付けを待たせることもできる。割込みマスク機能は、排他制御の実現や、割込み処理実行中に再び同一割込みが発生することを避けるために用いられる。

【００３７】

次に、このようなハードウェア上に実現される本発明のソフトウェア構成について説明する。図５に、事務処理ＯＳ１１０，リアルタイムＯＳ１１１の内部構成を示した。図中、全てのコンポーネントはメモリ１０１上に格納されている。事務処理ＯＳ，リアルタイムＯＳとも、実行可能タスクの情報を待ち行列１５０，１５１の形で管理する。実行可能待ち行列１５０，１５１は優先順位毎に設け

られる場合もあるが、本実施例では、全ての実行可能なタスクが一つの待ち行列によって管理される。なお、実行可能待ち行列が一つであっても、優先順位毎に設けられていても、本発明の内容には影響を与えない。各タスクは、（ａ）実行状態、（ｂ）実行可能状態、（ｃ）待ち状態の三種の状態を順次とるのでオペレーティングシステムは、実行可能タスク以外に、待ち状態タスク，停止状態タスクなどを別の待ち行列で管理する。尚、図５ではこれらの待ち行列の記載を省略している。実行可能待ち行列１５０および１５１によって管理されるタスク管理テーブル１６０～１６５は、実行すべきタスクの優先順位，タスク実行時のプログラムカウンタ・ステータスレジスタ・汎用レジスタの値などを保存する。

【００３８】

実行可能タスクを一つの待ち行列で管理する場合、実行可能待ち行列に登録されるタスク管理テーブルは優先順位の高い順に配置される。すなわち、次に実行すべきタスクの管理テーブルが待ち行列の先頭に来るように構成する。前述したように、本実施例で述べる事務処理ＯＳ１１０は、０～３１までの優先順位を有し、優先順位の数値が大きいほど優先順位が高い。また、リアルタイムOS111 は、０～２５５までの優先順位を有し、優先順位の数値が小さいほど優先順位が高い。このため、事務処理ＯＳでは、タスク１１４（優先順位「３１」）に対応する管理テーブル１６２が実行可能待ち行列１５０の先頭に配置され、以下、タスク１１３（優先順位「7」）の管理テーブル１６１，タスク１１２（優先順位「０」）の管理テーブル１６０の順に配置される。逆に、リアルタイムＯＳでは、タスク１１５（優先順位「０」）に対応する管理テーブル１６３が実行可能待ち行列１５１の先頭に配置され、次に、タスク１１６（優先順位「９９」）の管理テーブル１６４，タスク１１７（優先順位「２５５」）の管理テーブル１６５の順に配置される。

【００３９】

各オペレーティングシステムは、また、割込み処理を行う割込みハンドラ152，１５３，タスクに対してサービスを提供するシステムコールプログラム１５６，１５７、および、タスク切り替えを行うリスケジューラ１５４，１５５を有する。タスクの生成・削除・停止・再開や、外部割込み・内部割込み発生に伴ってタ

スク切り替えを行わなければならない場合、リスケジューラ１５４，１５５が起動される。リスケジューラ１５４，１５５は、直前に実行していたタスクの実行環境（レジスタなど）をタスク管理テーブルに格納した後に呼び出され、新たに実行すべきタスク決定して、その実行環境をタスク管理テーブルから取り出す。これをプログラムカウンタ，ステータスレジスタ，汎用レジスタなどに設定することにより、選択したタスクを実行する。

【００４０】

リスケジューラ１５４，１５５は、その内部に優先順位通知モジュール１２０，１２１を有する。優先順位通知モジュール１２０，１２１は、新たに実行すべきタスクの実行環境をレジスタに設定する直前に起動され、タスクの優先順位をオペレーティングシステム切り替えプログラム１１８内の優先順位変換モジュール１２２，１２３に通知する。

【００４１】

図６には、オペレーティングシステム切り替えプログラム１１８の内部構成を示したものである。優先順位変換モジュール１２２，１２３は、各オペレーティングシステムでの優先順位を正規化優先順位に変更するため、優先順位変換テーブル１７０，１７１をそれぞれ所有する。正規化優先順位は、本実施例では、０～２５５の範囲の整数であり、「正規化優先順位の数値が大きいほど、正規化優先順位が高い」と定義する。勿論、正規化優先順位の数値の範囲を変更したり、「正規化優先順位の数値が小さいほど、正規化優先順位が高い」と定義することも可能である。

【００４２】

前述したように、事務処理ＯＳ１１０は０～３１までの優先順位を有している。このとき、優先順位変換テーブル１７０は、３２個のエントリを有する配列となる。各配列要素は０～２５５の範囲の整数を有し、かつ、以下の不等式を満足する（ここでは、配列の名称をprioBusinessとする）。

【００４３】

$i > j \Leftrightarrow prioBusiness[i] > prioBusiness[j]$

（ただし、$0 \leqq i, j \leqq 31$）

事務処理ＯＳ１１０の優先順位、および、正規化優先順位ともに、数値が大きいほど優先順位が高いとしているため、この条件が成立する必要がある。

【００４４】

同様に、リアルタイムＯＳ１１１が０～２５５までの優先順位を有している場合、優先順位変換テーブル１７１は、２５６個のエントリを有する配列となる。各配列要素は０～２５５の範囲の整数を有し、かつ、以下の不等式を満足する（配列の名称をprioRealtimeとする）。

【００４５】

ｉ＞ｊ⇔prioRealtime［ｉ］＜prioRealtime［ｊ］

（ただし、０≦ｉ，ｊ≦２５５）

事務処理ＯＳと不等号の向きが異なる理由は、リアルタイムＯＳでは、優先順位の数値が小さいほど優先順位が高いとしているためである。

【００４６】

図６では、事務処理ＯＳ１１０上でタスク１１４が次に実行可能であると判定された場合、優先順位通知モジュール１２０から優先順位の数値「３１」が通知される。優先順位変換モジュール１２２は、優先順位変換テーブル１７０を利用し、正規化優先順位の数値「１２４」を入手することができる。逆に、リアルタイムＯＳ１１１上でタスク１１５が次に実行可能となった場合、優先順位通知モジュール１２１からリアルタイムＯＳでの優先順位の数値「０」が通知される。優先順位変換モジュール１２３は、優先順位変換テーブル１７１を使用して、正規化優先順位の数値「２５５」を入手する。ここで、本実施例では、事務処理ＯＳ１１０のタスクの優先順位がいくら高くても、正規化優先順位の数値が「１２４」を越えないことに注意する必要がある。これは、優先順位変換テーブルの設定方法の極端な例であり、リアルタイムＯＳ上のタスクをできる限り優先して実行させるという目的のためのものである（リアルタイムＯＳ上のタスクでも、優先順位が低いものに対しては、事務処理ＯＳ上のタスクが優先することがある）。しかし、勿論、二つのオペレーティングシステムの優先順位ができるだけ対等に正規化優先順位に変換されるように、優先順位変換テーブル１７０の構成を変更することも可能である。

【００４７】

優先順位変換モジュール１２２，１２３が変換した正規化優先順位は、いずれも、優先順位比較モジュール１２４に通知される。優先順位比較モジュール124は、通知された事務処理ＯＳ正規化優先順位１７２とリアルタイムＯＳ正規化優先順位１７３とを保持する。この場合、事務処理ＯＳ正規化優先順位１７２の数値は「１２４」であり、リアルタイムＯＳ正規化優先順位１７３の数値は「255」である。

【００４８】

前述したように、優先順位通知モジュール１２０，１２１はリスケジューラ内に存在し、タスク切り替え時に実行される。タスク切り替えから次のタスク切り替えまでの期間は一つのタスクしか実行されないため、タスクの優先順位が動的に変化しない限り、この間、当該オペレーティングシステムに対する正規化優先順位も変化しない。優先順位通知モジュール１２０，１２１は、タスク切り替え時に必ず優先順位を通知してくるため、優先順位比較モジュール１２４内に保持している事務処理ＯＳ正規化優先順位１７２とリアルタイムＯＳ正規化優先順位１７３が、実際の各オペレーティングシステムでのタスクの優先順位を反映した数値となる。

【００４９】

優先順位比較モジュール１２４は、保持している各オペレーティングシステムの正規化優先順位１７２，１７３の数値を相互に比較して、より高い正規化優先順位を有するオペレーティングシステムを優先的に実行させる。図６の例では、より高い正規化優先順位を有するリアルタイムＯＳ１１１を実行させることになる。

【００５０】

オペレーティングシステム切り替えプログラム１１８は、優先順位変換モジュール１２２，１２３，優先順位比較モジュール１２４の他に、発生した割込みを各オペレーティングシステムに振り分ける共通割込みハンドラ１７４，オペレーティングシステム間での協調処理を実行するＯＳ間通信機能モジュール１７５，二つのオペレーティングシステムの実行環境切り替えを行うＯＳコンテクスト切

り替えモジュール１７６，タスクの優先順位によって割込みマスクを変更する割込みマスクレベル計算モジュール１７７を有する。このプログラムの詳細は後述する。

【００５１】

図７に事務処理ＯＳ１１０におけるリスケジューラ１５４の処理フローを示した。リアルタイムＯＳ１１１のリスケジューラ１５５も同様の処理を行う。リスケジューラ１５４は、一般に、（１）タイマ割込み処理終了時、（２）外部割込み処理終了時、（３）システムコール実行時などにおいて、現在実行中のタスクが実行可能でなくなるか、実行中タスクより高い優先順位のタスクが実行可能となった場合、実行中タスクのレジスタをタスク管理テーブルに退避した後に起動されるモジュールである。リスケジューラ１５４を起動する可能性のあるシステムコールには、（ａ）タスクの生成・終了・停止・再開（例えば、自分より優先順位の高いタスクを生成した場合）、（ｂ）排他制御の実行・終了（排他制御待ち状態に移行した場合など）に加え、（ｃ）タスクの優先順位変更（自己の優先順位を低下させた場合など）、といったものがある。

【００５２】

リスケジューラ１５４は、第一に、実行可能待ち行列１５０から最高優先順位のタスク管理テーブルを取り出す（処理１８１）。ここで、全てのタスクが待ち状態や停止状態となっていて、実行可能タスクが存在しないこともある。これを処理１８２で判定する。

【００５３】

実行可能タスクが存在しなければ、優先順位変換モジュール１２２にidle状態であることを通知する（処理１８４）。このとき、実行可能タスクが存在しないのであるから、idleループへ移行する（処理１８６）。idleループとは、実行すべき処理が現れるまで何も行わないプログラムである。ここで注意すべきことは、処理１８４において、優先順位変換モジュール１２２に対してidle状態を通知したことである。事務処理ＯＳ１１０がidle状態であれば、優先順位比較モジュール１２４はリアルタイムＯＳ１１１を優先して実行させる。したがって、両オペレーティングシステムが同時にidle状態に移行した場合を除いて、実際に処理

１８６でidleループが実行されることはない。再び事務処理ＯＳ１１０が実行される場合には、何らかの実行すべき処理が存在するのであるから、直ちにidleループを抜けて処理１８１に移行することになる。

【００５４】

処理１８２で実行すべきタスクが存在すると判定した場合、次に実行すべきタスクの優先順位をタスク管理テーブルから取り出して、優先順位変換モジュール１２２に通知する。通知した優先順位がリアルタイムＯＳ１１１の優先順位よりも低かった場合（正規化優先順位による比較）、この時点で、リアルタイムＯＳ１１１に切り替えられることになる。リアルタイムＯＳ１１１から逆に事務処理ＯＳ１１０へ切り替えが行われる場合、割込み処理などによって、実行しようとしているタスクが実行可能でなくなったり、更に高い優先順位のタスクが実行可能となったりしない限り、処理１８３の直後から再実行することになる（処理１８１で選択したタスクを起動する）。なお、当然のことながら、現在実行中のタスクが実行可能でなくなるか、実行中タスクより高い優先順位のタスクが実行可能となった場合には、リスケジューラ１５４自体が再起動される。処理185で、タスク管理テーブルからレジスタを復帰してプロセッサ１００のレジスタに登録することにより、選択したタスクの処理を実行する。

【００５５】

ここで、リスケジューラ１５４中、処理１８３，１８４が優先順位通知モジュール１２０に相当する。リスケジューラ１５５内にも同様に優先順位通知モジュール１２１が埋め込まれる。

【００５６】

次に、図８に、事務処理ＯＳ１１０に対応する優先順位変換モジュール１２２の処理フローを示す。優先順位変換モジュール１２２は優先順位通知モジュール１２０から呼び出され、直ちに、優先順位変換処理を実行する。最初に、優先順位通知モジュール１２０がidle状態を通知してきたか否かを判定する（処理190）。idle状態はあらゆる優先順位よりも低い優先順位を有すると見なすこともできるので、ここでは、いずれの正規化優先順位よりも低くなるように、（－１）という正規化優先順位の数値を有していると考える。処理１９３では、この数値

（－１）を優先順位比較モジュール１２４に通知する。idle状態でなければ、優先順位変換テーブル１７０から対応するエントリを読み出し（処理１９１）、これを優先順位比較モジュール１２４に通知する（処理１９２）。リアルタイムＯＳ１１１に対応する優先順位変換モジュール１２３の処理も同様である。

【００５７】

図９は優先順位比較モジュール１２４の処理フローである。優先順位比較モジュール１２４は、第一に、事務処理ＯＳ１１０からの正規化優先順位であるか、リアルタイムＯＳ１１１からの正規化優先順位であるかを判定する(処理２００)。事務処理ＯＳ１１０の正規化優先順位であれば、事務処理ＯＳ正規化優先順位１７２に入手した正規化優先順位を格納する（処理２０１）。リアルタイムＯＳ１１１からの正規化優先順位であれば、リアルタイムＯＳ正規化優先順位１７３に当該正規化優先順位を記憶する（処理２０２）。

【００５８】

正規化優先順位が通知されてくるのは、いずれか一方の正規化優先順位が変化した可能性がある場合のみであるため、ここで、事務処理ＯＳ正規化優先順位１７２とリアルタイムＯＳ正規化優先順位１７３の数値を比較する(処理２０３)。本実施例では、正規化優先順位の数値が大きいオペレーティングシステムを優先的に起動するとしている。したがって、事務処理ＯＳ１１０の正規化優先順位の数値がリアルタイムＯＳ１１１の正規化優先順位の数値より大きければ、次に実行すべきオペレーティングシステムを事務処理ＯＳ１１０とする(処理２０４)。これ以外の場合には、次にリアルタイムＯＳ１１１を実行させるものとする(処理２０５)。なお、ここでは、事務処理ＯＳ１１０とリアルタイムＯＳ１１１の正規化優先順位が等しくても、リアルタイムＯＳ１１１を実行させる。これを変更し、正規化優先順位が互いに等しい場合には事務処理ＯＳ１１０を実行させるとしても問題はない。また、優先順位変換テーブル１７０，１７１の内容を設定するにあたって、各オペレーティングシステムから得られる正規化優先順位が互いに等しくならないようにすることも、システムを単純化させるための一つの方法である。

【００５９】

次に処理２０６で、新たに実行すべきオペレーティングシステムが今まで実行していたオペレーティングシステムに等しいか否かを判定する。今までと異なったオペレーティングシステムを実行させる場合、ＯＳコンテクスト切り替えモジュール１７６に依頼して、オペレーティングシステム実行環境の退避・復帰を行わせる（処理２０７）。

【００６０】

図１０に割込みハンドラ１５２，１５３，共通割込みハンドラ１７４，ＯＳ間通信機能モジュール１７５，ＯＳコンテクスト切り替えモジュール１７６の詳細を示す（割込みマスクレベル計算モジュール１７７については、別の図面で詳細を説明する）。

【００６１】

割込みハンドラ１５２，１５３は、割込み発生時のレジスタなどの保存、一時変数の保持を行うための領域として、それぞれ、割込みスタック２１４，２１５を所有する。割込み発生時には、割込み発生直前のレジスタ値を保存し、このレジスタ値を割込み処理終了後に復帰してやらなければならない。このため、割込みスタック２１４，２１５が必要とされる。使用するプロセッサ１００の種類によっては、割込み発生時に自動的にレジスタが切り替わり、かつ、割込み処理終了後に切り替え前のレジスタに戻す機能を備えているものもある。しかしながら、多重割込みが可能なシステムを考慮した場合には、このようなハードウェアを使用していても、割込みスタックが必要となる（割込み処理中により高い緊急性を有する割込みが発生した場合、新たに発生した割込み処理を優先的に実行し、次いで、元の割込み処理に戻る必要がある）。

【００６２】

割込みハンドラ１５２，１５３は、更に、割込みスタック２１４，２１５中、どの領域まで使用したかを示すためのポインタとして、割込みスタックポインタ２１６，２１７を所有する。割込みハンドラ１５２，１５３は、割込みスタック、および、割込みスタックポインタを用いて割込み発生前の実行環境（レジスタなど）を保存し、必要な割込み処理を実行する。割込み処理終了後、割込み発生

前の実行環境を復帰して元々動作していたプログラムの実行を継続させる。

【００６３】

共通割込みハンドラ１７４は、発生した割込みを割込みハンドラ１５２，153に対して振り分ける役割を有する。このため、共通割込みハンドラ１７４は、現在実行しているオペレーティングシステムが事務処理ＯＳ１１０であるかリアルタイムＯＳ１１１であるかを記憶する領域として、実行ＯＳ記憶変数２１０を所有する。例えば、図１０に示した時点で事務処理ＯＳが処理を実行しているものと仮定すると、この場合、実行ＯＳ記憶変数２１０には、「事務処理ＯＳ」が記憶されている。勿論、実行ＯＳ記憶変数２１０が文字列「事務処理ＯＳ」を記憶することは非常に非効率的であるため、例えば、

・事務処理ＯＳ→０

・リアルタイムＯＳ→１

といった整数形式で記憶するとよい。共通割込みハンドラ１７４は、更に、発生した割込みがいずれのオペレーティングシステムに対応するかを示す割込み対応テーブル２１１を有する。割込み対応テーブル２１１は、個々の割込みがどちらのオペレーティングシステムで処理されるべきかを示す対応表である。本実施例では、図４に示したように、３２種類の割込み要因が存在する。このため、割込み対応テーブル２１１も３２個のエントリ(０～３１)から構成される。図１０の例では、割込み要因０が事務処理ＯＳで、割込み要因１がリアルタイムＯＳで，…，割込み要因３１がリアルタイムＯＳでというように、いずれのオペレーティングシステムで処理が実行されるべきかが定義されている。なお、効率上、割込み対応テーブル２１１の内容も、実行ＯＳ記憶変数２１０と同様、文字列形式ではなく、整数形式で記憶すべきである。

【００６４】

さて、計算機の種類によっては、一つの入出力機器を二つのオペレーティングシステムが共用することもある（例えば、事務処理ＯＳ１１０，リアルタイムＯＳ１１１の両者がディスプレイに文字や映像を出力する場合）。このようなシステムを実現するためには、割込み振り分け先のオペレーティングシステムを動的に変更できることが必要となる。本実施例で、振り分け先オペレーティングシ

ステムを固定化するのではなく、割込み対応テーブル２１１から入手しているのは、このためである。割込み対応テーブル２１１の内容を入出力機器の使用状況によって変更することにより、割込みの振り分け先を動的に変更することができる。

【００６５】

共通割込みハンドラ１７４は、実行ＯＳ記憶変数２１０を用いて現在実行しているオペレーティングシステムの種類を判定し、これが割込み対応テーブル211から得られた振り分け先オペレーティングシステムに一致しなければ、ＯＳコンテクスト切り替えモジュール１７６に切り替えを依頼する。オペレーティングシステムが一致するか、オペレーティングシステム切り替え終了後、該当するオペレーティングシステムの割込みハンドラに処理を依頼する。なお、実行ＯＳ記憶変数２１０はＯＳコンテクスト切り替えモジュール１７６からも参照される。このように、オペレーティングシステム切り替えプログラム１１８内の各モジュールが有する内部構造は、個々のモジュールによって占有されるものとは限らず、各モジュールが必要に応じて共用できる。

【００６６】

ＯＳコンテクスト切り替えモジュール１７６は、優先順位比較の結果、若しくは、割込み発生の結果、オペレーティングシステムを切り替えなければならない場合に起動される。二つのオペレーティングシステムを切り替えるため、これらの実行環境（すなわち、レジスタ値）を保存しておくための領域を所有しなければならない。ここでは、事務処理ＯＳの実行環境を保存するための領域として、事務処理ＯＳ用保存コンテクスト２１２，リアルタイムＯＳの実行環境を保存するための領域として、リアルタイムＯＳ用保存コンテクスト２１３を用意する。ＯＳコンテクスト切り替えモジュール１７６は、現在実行しているオペレーティングシステムに対応する保存コンテクストに実行環境を保存し、次に、もう一方の保存コンテクストから実行環境を読み出して、プロセッサ１００のレジスタに設定する。これにより、オペレーティングシステムの切り替えを実施できる。

【００６７】

ＯＳ間通信機能モジュール１７５は、二つのオペレーティングシステム上のタ

スクが相互に通信・協調するためのプログラムである。ここでは、事務処理ＯＳ１１０，リアルタイムＯＳ１１１の両者が使用できる共有メモリ２１８、並びに、オペレーティングシステム間で排他制御を行うためのロック取得モジュール２１９，ロック解放モジュール２２０が存在するとしている。共有メモリ218は、メモリ１０１の一部であり、両オペレーティングシステムから参照を行える領域である。共有メモリ２１８以外のメモリは、基本的に、事務処理ＯＳ用領域，リアルタイムＯＳ用領域，オペレーティングシステム切り替えプログラム用領域に分割され、それぞれ、事務処理ＯＳ１１０，リアルタイムＯＳ１１１，オペレーティングシステム切り替えプログラム１１８に占有されて使用される。二つのオペレーティングシステム上のタスクが共有メモリ２１８を使用する場合、通常、排他制御を行って共有メモリ２１８上のデータの一貫性を保つ。アプリケーションプログラムがオペレーティングシステム間に跨る排他制御を行う場合、ロック取得モジュール２１９，ロック解放モジュール２２０が提供する機能を利用する。

【００６８】

このとき、優先順位逆転現象と呼ばれる状況について注意しておかなければならない。優先順位逆転現象とは、一般に、以下の状態を意味する。

【００６９】

（１）低優先順位であるタスクαが最初に起動されてロックを取得する。

【００７０】

（２）次に高優先順位であるタスクβが起動されてロック待ち状態に移行する。

（３）次に中優先順位であるタスクγが起動され、タスクα（低優先順位）からタスクγ（中優先順位）への切り替えが行われる。

【００７１】

このとき、タスクγの実行によってタスクαの実行が妨げられ、結果的に、高優先順位タスクβがロックを取得するまでの時間が延びてしまうことになる。優先順位逆転現象は、通常、優先順位上昇方式または優先順位継承方式を用いて解決する。

【００７２】

優先順位上昇方式は、ロック取得を行ったタスクの優先順位を一定のレベルまで高くする方式である。これによって、ロック取得を行ったタスクαの優先順位が一時的にタスクγ（中優先順位）よりも高くなり、タスクγが起動されても、ロック解放まではタスクαが実行される。この後、高優先順位であるタスクβがロックを取得し、処理を継続することになる。

【００７３】

優先順位継承方式は、前記優先順位上昇方式を変更して柔軟性を持たせたものである。優先順位継承方式では、ロック待ち状態に移行したタスク（例では、タスクβ）の優先順位がロックを取得しているタスク（タスクα）よりも高ければ、この優先順位をロック取得中タスクに継承させる方式である。ここでは、ロック取得中の間だけ、タスクαがタスクβの優先順位を受け継ぐことになる。低優先順位のタスク同志で排他制御を行う場合など、必ずしも優先順位を一定レベルまで上昇させる必要がない場合もあり、優先順位継承方式は、このような状況にも対処できる方式である。

【００７４】

本実施例で述べるロック取得モジュール２１９，ロック解放モジュール２２０は、このような優先順位上昇もしくは優先順位継承方式をオペレーティングシステム間で実施できるものとする。オペレーティングシステム間での優先順位上昇・継承は、正規化優先順位を用いて行われる。

【００７５】

以降、図１０で示したモジュールのうち、ＯＳコンテクスト切り替えモジュール１７６，共通割込みハンドラ１７４，割込みハンドラ１５２，ロック取得モジュール２１９，ロック解放モジュール２２０の処理フローの説明を行う。割込みハンドラ１５３に関しては、割込みハンドラ１５２と同様の処理フローを有するため、説明を省略する。

【００７６】

図１１はＯＳコンテクスト切り替えモジュール１７６の処理フローである。

【００７７】

ＯＳコンテクスト切り替えモジュール１７６は、オペレーティングシステムを切り替えなければならない場合にのみ呼び出され、切り替えが必要か否かのチェックは呼出し前に行われている。したがって、最初に、どちらのオペレーティングシステムへ切り替えなければならないかのみをチェックする（処理２３０）。ここで、事務処理ＯＳ１１０からリアルタイムＯＳ１１１への切り替えを行う場合（現実行オペレーティングシステムが事務処理ＯＳ１１０であって、リアルタイムＯＳ１１１の優先順位が高くなった場合）、事務処理ＯＳ用保存コンテクスト２１２内に使用中レジスタの値を退避する（処理２３１）。次に、リアルタイムＯＳ用保存コンテクスト２１３から実行環境を復帰して、レジスタに設定することにより（処理２３２）、以前退避した状態からリアルタイムＯＳ１１１を再実行させることができる。リアルタイムＯＳ１１１から事務処理ＯＳ１１０への切り替えを行う場合、逆に、リアルタイムＯＳ用保存コンテクスト２１３に使用中レジスタの値を退避し（処理２３３）、次に、事務処理ＯＳ用保存コンテクスト２１２からレジスタを復帰する（処理２３４）。いずれの場合においても、最後に、切り替え後のオペレーティングシステムの種類を実行ＯＳ記憶変数２１０に書き込む（処理２３５）。

【００７８】

図１２には、共通割込みハンドラ１７４の処理フローを示している。一般に、単一のオペレーティングシステムによって制御される計算機では、一旦、全ての割込みを割込みハンドラと呼ばれるモジュールが処理し、次いで、各プログラムに振り分ける。しかしながら、本実施例で説明するような複数個のオペレーティングシステムを動作させる計算機では、この更に前に、共通割込みハンドラが全ての割込みを受け付け、これを対応するオペレーティングシステムの割込みハンドラに振り分ける形となる。各オペレーティングシステムに割込みを振り分けるに当たって、割込み発生時に、割込み対象とは別のオペレーティングシステムが処理を実行していることもある。この場合、割込みに対応するオペレーティングシステムに切り替える必要があり、共通割込みハンドラはこのような役割も有することになる。

【００７９】

共通割込みハンドラ１７４は、割込み発生後、第一に、実行ＯＳ記憶変数210の内容を取り出して、現在実行中のオペレーティングシステムが事務処理OS110であるかリアルタイムＯＳ１１１であるかをチェックする（処理２４０）。次に、割込み対応テーブル２１１を用いて、発生した割込みがいずれのオペレーティングシステムに対応しているかを入手する（処理２４１）。例えば、図１０の割込み対応テーブル２１１を用いるとすると、割込み「０」が発生した場合には事務処理ＯＳ１１０，割込み「１」が発生した場合にはリアルタイムＯＳ１１１，…，割込み「３１」が発生した場合にはリアルタイムＯＳ１１１、というように対応するオペレーティングシステムを入手することができる。ここで、割込み対象オペレーティングシステムが実行中オペレーティングシステムに等しいか否かをチェックする（処理２４２）。

【００８０】

発生した割込みが現在実行中のオペレーティングシステムに対するものでなければ、一旦、オペレーティングシステムの切り替えを行わなければならない。この切り替え処理は、ＯＳコンテクスト切り替えモジュール１７６に依頼することによって行う（処理２４３）。次に割込み対象オペレーティングシステムが事務処理ＯＳ１１０であるかリアルタイムＯＳ１１１であるかをチェックし（処理２４４）、対象が事務処理ＯＳ１１０であれば、事務処理ＯＳの割込みハンドラ１５２を起動する（処理２４５）。リアルタイムＯＳ１１１への割込みが発生していれば、リアルタイムＯＳの割込みハンドラ１５３を起動することになる（処理２４６）。一般に、割込みハンドラ１５２，１５３は、割込み処理によってタスク切り替えを行わなければならないとき、リスケジューラ１５４，１５５を呼び出し、共通割込みハンドラ１７４に制御を戻すことはない。しかし、タスク切り替えが発生しない場合、そのまま割込みハンドラの処理を終了する。このとき、割込みハンドラ１５２，１５３は、共通割込みハンドラへ制御を戻し（図１３で後述）、共通割込みハンドラは処理２４７から動作を再開する。処理２４７は、割込み発生時にオペレーティングシステムを切り替えたかどうかをチェックする処理である。処理２４３によってオペレーティングシステムを切り替えた場合

、ここで再び、オペレーティングシステムを切り替え、元の実行環境に戻す必要がある。タスク切り替えが発生しないということは、両オペレーティングシステムの正規化優先順位が変化しなかったということを意味し、割込み発生前の実行状態に戻さなければならないわけである。このため、もう一度ＯＳコンテクスト切り替えモジュール１７６に依頼して、切り替え処理を実行させる(処理２４８)。

【００８１】

各オペレーティングシステムの割込みハンドラがリスケジューラ１５４，155を実行した場合、前述したように、共通割込みハンドラの処理２４７に制御が戻ることはない。この場合、リスケジューラ１５４，１５５が新たなタスクの優先順位を優先順位変換モジュール１２２，１２３に通知してくる。これによって、優先順位比較モジュール１２４がオペレーティングシステムの切り替えを行うか否かの判定を行う。

【００８２】

なお、共通割込みハンドラ１７４の構成を図２８のように変更することも可能である。図２９に示した共通割込みハンドラ１７４は、図１０に示した共通割込みハンドラの構成要素の他に、割込み優先順位対応テーブル３８０を所有する。割込み優先順位対応テーブル３８０は、割込みハンドラがいかなる正規化優先順位で動作しなければならないかをしめす対応表である。図２９の例では、割込み要因０に対応する割込みハンドラは、正規化優先順位２５５で動作しなければならず、また、割込み要因３１に対応する割込みハンドラは、正規化優先順位224で動作しなければならない。共通割込みハンドラ１７４は各オペレーティングシステムの割込みハンドラを起動する際に、割込み優先順位対応テーブル３８０にしたがって、当該オペレーティングシステムの正規化優先順位を更新し、また、割込みハンドラ終了後、元の正規化優先順位を復帰させる。なお、この例では共通割込みハンドラ１７４が割込み優先順位対応テーブル３８０を有するものとしているが、割込み優先順位対応テーブル３８０を優先順位変換モジュール１２２，１２３内に設け、共通割込みハンドラ１７４から優先順位変換を依頼する構成を採ることも可能である。

【００８３】

なお、図２９のように、割込みに対しても正規化優先順位を割当てることができるのならば、各オペレーティングシステムのあらゆる動作状態にも正規化優先順位を割当てることができる。例えば、オペレーティングシステムは、一般に以下の四種類の動作状態を持つ。

【００８４】

（１）idle状態

（２）タスク実行状態

（３）オペレーティングシステム自体の処理状態（例えば初期化処理）

（４）割込み処理状態

これら全てに正規化優先順位を割当てることも可能である。この場合、一般に考えて、割込み処理状態が最も優先して動作しなければならず、最も高い正規化優先順位を割当てる。次にオペレーティングシステム自体の処理状態，タスク実行状態，idle状態の順に正規化優先順位を割当てていくことになる。

【００８５】

図１３は共通割込みハンドラ１７４から呼び出される事務処理ＯＳの割込みハンドラ１５２の処理フローを示したものである。第一に、使用中レジスタを割込みスタック２１４に退避し（処理２５０）、割込み処理を実行する(処理２５１)。ここで、オペレーティングシステムがリスケジューリング中か否か（すなわち、リスケジューラ１５４の処理実行中か否か）をチェックする（処理２５２）。リスケジューリング中とは、次に実行すべきタスクを選択している最中であり、実際にはタスクを実行していないことを意味する。したがって、オペレーティングシステムの処理を単純化させるならば、この場合、再びリスケジューリングを最初からやり直せば良いことになる。すなわち、リスケジューリング中であると判定した場合、割込みスタックの退避レジスタを破棄し（処理２５７）、リスケジューラ１５４を最初から起動する（処理２５８）。なお、リスケジューリング中に割込みが発生しても、実行すべきタスクを新たに選択し直す必要がない場合もある。本方式は、この状況でもリスケジューリングを最初から行うことになり、効率的でないこともある。このため、リスケジューリング中に発生し、リスケ

ジューラ１５４の実行に影響を与える処理（例えば、タスク起動・終了など）を待ち行列などに登録しておく方法を用いることもできる。この場合、リスケジューラ１５４の終了前や優先順位通知モジュール１２０内で、該待ち行列に登録しておいた処理を纏めて実行する。この方式を採用すると、割込み発生のたびに途中まで実行していたリスケジューリングをやり直す必要が無い。しかしながら、当該待ち行列に登録していた処理を纏めて実行した後、これによってリスケジューリングが再び必要となるか否かを再チェックしなければならない。

【００８６】

なお、本実施例では、単純化のため、前者の方式で説明を行う。しかし、後者の方式を用いる場合、

（ａ）リスケジューリング中か否かを示すフラグ

（ｂ）処理待ち行列

を用意すればよい。オペレーティングシステムが提供するシステムコールなどにおいて、リスケジューラ１５４の実行に影響を与える処理を、リスケジューリング中であれば、該待ち行列に登録する。更に、リスケジューラ１５４の処理フロー中に、該待ち行列に登録された処理を一括して実行するモジュールを挿入する。

【００８７】

割込みハンドラ１５２がリスケジューリング中でないと判定した場合、当該オペレーティングシステムはその時点で何らかのタスクを実行している。このため、第一に、タスク切り替えが必要か否かを判定する（処理２５３）。タスク切り替えが必要なければ（現実行タスクが最高優先順位であり、かつ、実行可能であれば）、このまま、割込みハンドラ１５２の処理を終了する。このため、割込みスタック２１４上に退避しておいたレジスタ値を復帰し（処理２５４）、共通割込みハンドラへ制御を戻す（処理２５５）。タスク切り替えが必要であると判定した場合、割込みスタック２１４上に退避したレジスタの値をタスク管理テーブルにコピーし（処理２５６）、割込みスタック上の退避レジスタを破棄する（処理２５７）。この後、リスケジューラ１５４を起動する（処理２５８）。なお、プロセッサ１００に、メモリ１０１上データの参照を行って、同時に、割込みス

タックポインタ２１６の値を増減できる機能が備わっていれば、処理２５６と処理２５７を纏めて実行することができる。

【００８８】

次に、ロック取得モジュール２１９，ロック解放モジュール２２０の処理フローを説明する。ここでは、優先順位上昇方式を用いる例を説明し、優先順位継承方式を使用する実施例については後述する。なお、ここでは、ロック取得・解放を要求したタスク、および、それを管理するオペレーティングシステムを自タスク，自オペレーティングシステムと呼び、もう一方のオペレーティングシステムを別オペレーティングシステムと呼んでいる。

【００８９】

図１４が優先順位上昇方式を用いるロック取得モジュール２１９の処理フローである。ロック取得モジュール２１９は、第一に、別オペレーティングシステムのタスクがロック取得中か否かをチェックし（処理２６０）、取得中でなければ、自オペレーティングシステムがロックの所有者であるように設定する（処理２６１）。別オペレーティングシステムのタスクがロック取得中であれば、ロックが解放されるまでタスクを待ち状態に移行させる（処理２６３）。このとき、別オペレーティングシステムを優先して実行させ、当該ロックを速やかに解放させるようにしなければならない。このため、優先順位比較モジュール１２４内に記憶されている別オペレーティングシステムの正規化優先順位を自オペレーティングシステムより高くなるように変更する（処理２６２）。ここで処理２６３においてタスクを待ち状態に移行させると、自オペレーティングシステムでリスケジューラが起動される。これにより、リスケジューラから、優先順位通知モジュール，優先順位変換モジュール，優先順位比較モジュールと順次呼び出され、正規化優先順位を高く変更した別オペレーティングシステムに切り替えが行われることになる。

【００９０】

図１５が優先順位上昇方式を用いる場合のロック解放モジュール２２０の処理フローである。第一に、自己が所有するロックを解放する（処理２７０）。次に、別オペレーティングシステムがロック取得待ちか否かをチェックする（処理

２７１）。ロック取得待ちでなければ、正規化優先順位の上昇は行われていない。このため、このままロック解放モジュール２２０の処理を終了する。しかし、別オペレーティングシステムがロック取得待ちであれば、優先順位逆転現象を解決するため、正規化優先順位の上昇が行われている筈である。したがって、自オペレーティングシステムの正規化優先順位を元の値に戻し（処理２７２）、次に、別オペレーティングシステムでロック取得待ちとなっているタスクを実行可能状態に移行させる（処理２７３）。ロック取得待ちタスクを実行可能とすることによりタスク切り替えが発生すれば、別オペレーティングシステムのリスケジューラが起動され、前述のロック取得モジュールの場合と同様、最終的に、優先順位比較モジュール１２４が実行される。これにより、元の正規化優先順位を用いてオペレーティングシステムの重要性が比較され、より高い正規化優先順位を有するオペレーティングシステムが選択される。

【００９１】

次に、優先順位継承方式を用いる例を説明する。この場合、図１６に示すように、各オペレーティングシステムは、正規化優先順位にしたがってタスクの優先順位を変更する優先順位設定モジュール２８０，２８１を装備しなければならない。また、優先順位変換モジュール１２２，１２３は、優先順位を正規化優先順位に変換する機能のほか、正規化優先順位を優先順位に変換する機能を搭載しなければならない。このため、優先順位変換モジュール１２２，１２３は、正規化優先順位をインデクスとし、各オペレーティングシステムの優先順位を内容とする優先順位逆変換テーブル２８２，２８３を有する。

【００９２】

オペレーティングシステムは、一般に、システムコールとして、タスクの優先順位を変更する機能を有している。優先順位設定モジュール２８０，２８１は、与えられた正規化優先順位を各オペレーティングシステムに対応する優先順位に変換し、タスク優先順位変更システムコールなどを用いてタスクの優先順位を変更する。優先順位逆変換テーブル２８２，２８３は、優先順位設定モジュール２８０，２８１の機能をサポートする役割を有する。本実施例では、正規化優先順位が０～２５５であるため、優先順位逆変換テーブル２８２，２８３は、256

個のエントリを有する配列である。また、事務処理ＯＳ１１０の優先順位の範囲が０～３１であるため、優先順位逆変換テーブル２８２の各配列要素は、０～３１の範囲のいずれかの整数を有し、かつ、以下の不等式を満足する（配列の名称をrevBusinessとする）。

【００９３】

$i > j \Rightarrow revBusiness[i] \geqq revBusiness[j]$

（ただし、$0 \leqq i, j \leqq 255$）

リアルタイムＯＳ１１１は優先順位０～２５５を有し、かつ、優先順位の数値が小さいほど優先順位が高いという特徴を有するため、優先順位逆変換テーブル２８３の各配列要素は、０～２５５までのいずれかの整数を有し、かつ、以下の不等式を満足する（配列の名称をrevRealtimeとする）。

【００９４】

$i > j \Rightarrow revRealtime[i] \leqq revRealtime[j]$

（ただし、$0 \leqq i, j \leqq 255$）

さて、本実施例では、正規化優先順位の数値が０～２５５、リアルタイムＯＳの優先順位の数値が０～２５５であるとしているため、正規化優先順位とリアルタイムＯＳの優先順位との対応付けを一対一に行うことができている。したがって、優先順位変換テーブル１７１から一意に優先順位逆変換テーブル２８３を導出することができる。しかしながら、事務処理ＯＳの優先順位の数値は０～３１であるため、優先順位逆変換テーブル２８２を一意に定めることはできない。この場合、例えば、正規化優先順位「１」に対応する事務処理ＯＳの優先順位を、優先順位変換テーブル１７０から導き出すことができない。このため、一意に定めることのできない正規化優先順位に対しては、上記不等式を満たすように、事務処理ＯＳの優先順位を適宜に定めている。

【００９５】

優先順位変換モジュール１２２のうち、優先順位逆変換機能の処理フローについて図１７に示した。ここでは、指定正規化優先順位に対応する事務処理ＯＳの優先順位を優先順位逆変換テーブル２８２から取り出し（処理２９０）、当該優先順位を優先順位設定モジュール２８０に通知する（処理２９１）。優先順位変

換モジュール１２３も同様の優先順位逆変換機能を有する。

【００９６】

これに対する優先順位設定モジュール２８０の処理フローを図１８に示す。優先順位設定モジュール２８０は、第一に、優先順位変換モジュール１２２に依頼して、正規化優先順位から事務処理ＯＳの優先順位への変換を行わせる（処理２９２）。次に、事務処理ＯＳの優先順位変更システムコールなどを用いてタスクの優先順位を変更させる（処理２９３）。リアルタイムＯＳ１１１における優先順位設定モジュール２８１の処理も同様である。

【００９７】

次に、これらの優先順位逆変換機能，優先順位設定モジュールを用いた優先順位継承方式の実現方法について述べる。

【００９８】

図１９は優先順位継承方式を用いるロック取得モジュール２１９の処理フローである。ロック取得モジュール２１９は、第一に、別タスクがロック取得中か否かをチェックし（処理３００）、取得中でなければ、自タスクがロックの所有者であるように設定する（処理３０１）。別タスクがロック取得中であれば、ロックが解放されるまでタスクを待ち状態に移行させる必要がある（処理３０４）。このとき、ロック取得中タスクを優先して実行させ、当該ロックを速やかに解放させるようにしなければならない。ロック取得中タスクの正規化優先順位と自タスクの正規化優先順位を比較し（処理３０２）、ロック取得中タスクの正規化優先順位が低ければ、当該タスクに自タスクの優先順位を継承させる(処理３０３)。優先順位継承は、ロック取得中タスクを実行するオペレーティングシステムの優先順位設定モジュールに対し、自タスクの正規化優先順位を指定して優先順位設定を依頼することによって行うことができる。ロック取得中タスクの正規化優先順位が既に自タスクの正規化優先順位より高ければ、優先順位継承を行う必要はなく、直ちにタスクを待ち状態に移行させることができる。

【００９９】

図２０は優先順位継承方式を用いるロック解放モジュール２２０の処理フローである。第一に、自己が所有するロックを解放する（処理３１０）。次に、別タ

スクがロック取得待ちか否かをチェックする（処理３１１）。ロック取得待ちでなければ、正規化優先順位継承が行われていないことになり、このままロック解放モジュール２２０の処理を終了する。しかし、別タスクがロック取得待ちであれば、正規化優先順位継承が行われている筈である。したがって、優先順位設定モジュールに依頼して自タスクの正規化優先順位を元の値に戻し(処理３１２)、次に、ロック取得待ちとなっているタスクを実行可能状態に移行させる（処理３１３）。

【０１００】

本実施例では説明を省略したが、ロック取得モジュール２１９，ロック解放モジュール２２０などを実行している途中に割込みが発生し、タスク切り替えなどが起こると、ロック所有者を示すデータなどに不整合が生じる可能性がある。したがって、通常、これらの処理実行中には、割込みマスク機能によって割込み発生を抑止する。

【０１０１】

図１６～図２０に示したプログラムモジュール，処理フローにより、オペレーティングシステム間での排他制御において、優先順位継承方式を用いることが可能となる。なお、実際には、図１９，図２０に示した処理フローを用いることにより、一つのオペレーティングシステム内で正規化優先順位の継承を行うことも可能である。

【０１０２】

図２１に、割込みマスクレベル計算モジュール１７７の内部構造を示した。割込みマスクレベル計算モジュール１７７は、正規化優先順位を割込みマスクレベルに変換し、一定以上の正規化優先順位であれば、特定の割込みをマスクする機能である。これは、ある一定以上の正規化優先順位を有するタスクは、割込み処理よりも優先して実行されなければならないという考えに基づく機能である。したがって、システムコールなどで割込みをマスクする機能が備わっていれば、本機能が存在しなくても十分システムは成立する。ただし、割込みマスクレベル計算モジュール１７７の利点として、正規化優先順位が一定値以上になれば、自動的に割込みマスクを実行することがある。

【０１０３】

割込みマスクを実行するため、割込みマスクレベル計算モジュール１７７は、割込みマスク変換テーブル３２０を所有する。割込みマスク変換テーブル３２０は、正規化優先順位から割込みマスク値を入手するための変換テーブルである。本実施例では、図４に示したように、３２種類の割込み要因を個別にマスクできる計算機アーキテクチャを対象としているため、割込みマスク変換テーブル320の各エントリは３２ビット長データを有する。しかしながら、図３に示したように、４ビットの割込みマスクレベルを使用すると、割込みマスク変換テーブル３２０の各エントリを４ビット長に縮少させることができる。ここで、図２１に示した実施例では、例えば、正規化優先順位「０」に対応する割込みマスク値が0x00000000であり、正規化優先順位「２５５」に対応する割込みマスク値は0xffffffffである。これは、正規化優先順位が「０」であれば全ての割込みを許可し、最高の正規化優先順位でシステムが動作しているならば、全ての割込みをマスクすることを意味する。優先順位比較モジュール１２４が割込みマスクレベル計算モジュール１７７に対して、現在実行中オペレーティングシステムの正規化優先順位を通知することにより、必要な割込みマスクを自動的に行うことができる。

【０１０４】

図２２に割込みマスクレベル計算モジュール１７７の処理フローを示す。ここでは、第一に、正規化優先順位に対応する割込みマスク値を割込みマスク変換テーブル３２０から取り出す（処理３３０）。次に、入手した割込みマスク値を割込みマスクレジスタ１４３などに設定して、割込みマスクを実施する（処理331）。

【０１０５】

以上に本発明の第一の実施例について説明した。本実施例を用いることにより、各オペレーティングシステムが実行しているタスクの優先順位にしたがって、オペレーティングシステムの切り替えを行うことができる。本実施例では、各オペレーティングシステム内に優先順位通知モジュールを組み込み、リスケジューリングのたびに優先順位の変更を通知するように改造しなければならない。しか

しながら、市販オペレーティングシステムを使用する場合など、このような改造を施せないことも多い。以下、これに対処する本発明の第二の実施例について説明する。

【０１０６】

図２３は本発明の第二の実施例を示した構成図である。事務処理ＯＳ１１０，リアルタイムＯＳ１１１は、それぞれ、現在実行中タスクの優先順位（実行優先順位３４０，３４１）を保持するものとする。各オペレーティングシステムは、図５に示す実行可能待ち行列１５０，１５１の先頭に存在するタスク管理テーブル１６２，１６３内に格納されている優先順位の数値が実行優先順位３４０，３４１となる。また、実行可能待ち行列が優先順位毎に複数個存在する場合には、一般に、現時点で最高優先順位の実行可能待ち行列がどこに存在するかを示すポインタや、最高優先順位の数値そのものを示す変数（優先順位保持変数）を用意することが多い。前者の場合には、該ポインタから実行可能待ち行列を辿り、タスク管理テーブルを参照することにより、後者の場合には、該優先順位保持変数自体を読み出すことにより、実行優先順位３４０，３４１を入手できる。

【０１０７】

オペレーティングシステム切り替えプログラム１１８内には、新たに、優先順位監視モジュール３４４を設ける。これ以外のモジュールである、優先順位変換モジュール１２２，１２３，優先順位比較モジュール１２４，共通割込みハンドラ１７４，ＯＳ間通信機能モジュール１７５，ＯＳコンテクスト切り替えモジュール１７６，割込みマスクレベル計算モジュール１７７は、第一の実施例で説明した内容と同様の構成，処理フローを有する。各オペレーティングシステム内には、優先順位通知モジュール１２０，１２１は存在しない。

【０１０８】

優先順位監視モジュール３４４は、事務処理ＯＳ実行優先順位記憶値３４２とリアルタイムＯＳ実行優先順位記憶値３４３を所有する。優先順位監視モジュール３４４は定期的に起動され、各オペレーティングシステムの実行優先順位340，３４１を監視する。ここで、記憶している実行優先順位３４２または３４３と、現在の実行優先順位３４０または３４１が異なっていれば、優先順位が変化した

と認識して、優先順位変換モジュール１２２または１２３に新たな実行優先順位を通知する。

【０１０９】

優先順位監視モジュール３４４の処理フローを図２４に示す。第一に、事務処理ＯＳの実行優先順位３４０を読み出して、これを事務処理ＯＳ実行優先順位記憶値３４２と比較する（処理３５０）。これにより、事務処理ＯＳの優先順位が変化したか否かを認識できる。優先順位が変化していなければ何も実行せず、処理３５３に移行する。優先順位が変化していれば、事務処理ＯＳの実行優先順位３４０を事務処理ＯＳ実行優先順位記憶値３４２に登録し（処理３５１）、次に、当該優先順位を事務処理ＯＳの優先順位変換モジュール１２２に通知する（処理３５２）。リアルタイムＯＳ１１１の実行優先順位変化に対しても同様の処理を行う。第一に、リアルタイムＯＳの実行優先順位３４１とリアルタイムＯＳ実行優先順位記憶値３４３との比較を行い（処理３５３）、優先順位の変化が無ければ処理を終了する。優先順位が変化していれば、リアルタイムＯＳの実行優先順位３４１をリアルタイムＯＳ実行優先順位記憶値３４３に登録し(処理３５４)、これをリアルタイムＯＳの優先順位変換モジュール１２３に通知する（処理３５５）。

【０１１０】

なお、優先順位変換モジュールに優先順位を通知した結果、オペレーティングシステムの切り替えが行われると、優先順位監視モジュール３４４に制御が戻ってこないことに注意しなければならない。このため、図２４の監視方法は、常に事務処理ＯＳ１１０の優先順位変化を優先して監視するものであると言える。両オペレーティングシステムの監視を対等なものとするためには、優先順位監視モジュール３４４の起動回数を計数しておき、

（１）起動回数が奇数であれば事務処理ＯＳ１１０の優先順位変化を監視し、

（２）起動回数が偶数であればリアルタイムＯＳ１１１の優先順位変化を監視する

といった処理フローなどに改める必要がある。

【０１１１】

ここで、優先順位監視モジュール３４４は、「定期的に」起動され、優先順位の変化を監視することに注意しなければならない。定期的な処理を実現するためには、タイマ割込みによって優先順位監視モジュール３４４を起動すればよい。ここでは、タイマ割込み，外部割込みを含む全ての割込みが共通割込みハンドラ１７４で一括処理されることに着目し、共通割込みハンドラ１７４内に優先順位監視モジュール３４４を埋め込む方法について説明する。図２５が優先順位監視モジュール３４４を埋め込んだ共通割込みハンドラ１７４の処理フローである。図中、処理２４０～２４８は全て図１２に等しい。優先順位監視モジュール344の起動（処理３６０）はオペレーティングシステム切り替えを行う可能性がある。このため、割込みハンドラの実行（処理２４５，２４６）後に埋め込まなければならない。割込みハンドラ１５２，１５３が共通割込みハンドラ１７４に制御を戻すと、自動的に優先順位変化が監視され、必要に応じてオペレーティングシステムの切り替えが行われる。優先順位監視モジュール３４４がオペレーティングシステムの切り替えを行う必要がないと判断した場合、そのまま共通割込みハンドラ１７４の処理を終了する。

【０１１２】

なお、本発明の第二の実施例では、リスケジューリング直後の優先順位変化を優先順位変換モジュールに通知する手段が存在しない。すなわち、一定間隔毎に優先順位の変化をチェックするのであって、優先順位の変化に直ちに反応してオペレーティングシステムを切り替えることにはならない。このため、本発明の第二の実施例は、オペレーティングシステム内部の改造を伴わない計算機を実現できるが、第一の実施例に比べ、切り替え効率的に劣るという特徴がある。

【０１１３】

本発明の第二の実施例は、両オペレーティングシステムの内部を改造しないという条件を満たすものであるが、例えば、リアルタイムＯＳ１１１は改造できるが、事務処理ＯＳ１１０を改造することができない、といった条件が発生することもある。この場合、本発明の第三の実施例として、リアルタイムＯＳ１１１内部に優先順位通知モジュール１２１を搭載し、優先順位監視モジュール３４４は

事務処理ＯＳ１１０の優先順位変化のみを監視するといった計算機を実現することも可能である（図２６）。

【０１１４】

また、リスケジューラ内部に優先順位通知モジュールを組み込むのではなく、各オペレーティングシステムのタイマ割込み処理モジュール内部など、定期的に起動される部分に優先順位通知モジュールを組込み、優先順位監視モジュール３４４の代用とすることも可能である。

【０１１５】

以上説明した本発明を図２７に示す車載ナビゲーションシステムに適用した場合、処理時間の長い経路探索タスクが動作している場合でもユーザからの入力を受け付けることができる。つまり、この場合はインタフェースタスクに対して、経路探索タスクよりも高い正規化優先順位を与える。このとき、ユーザがボタンを押してインタフェースタスクが動作すれば、より正規化優先順位の高いタスクを動作させる事務処理ＯＳに切り替えが行われ、結果として、処理の長い経路探索タスクが動作している場合にもインターフェースタスクが実行されることになる。

【０１１６】

更に、複数のオペレーティングシステムを実行する仮想計算機においても、本発明を適用することにより異なった優先順位体系を有しているオペレーティングシステムを、実行すべきタスクの優先順位にしたがって実行させることができる。

【０１１７】

以上より、本発明によれば事務処理用ＯＳのユーザインタフェースとリアルタイムＯＳの信頼性を併せ持つ計算機を構築することができる。

【０１１８】

前記第一乃至第三の実施例は、全て、オペレーティングシステム切替えプログラム１１８の内部で優先順位の変換を行う構成を有している。しかしながら、優先順位変換モジュール１２２，１２３は、それぞれ、事務処理ＯＳ１１０，リアルタイムＯＳ１１１に個別に対応しているため、これらを、各オペレーティング

システム内部に搭載しても良い。このような変更を加えた第四の実施例を図２９に示す。

【０１１９】

本実施例では、第一に、優先順位変換モジュール１２２，１２３が各オペレーティングシステムでの優先順位を正規化優先順位に変換する。これを受けて、優先順位通知モジュール１２０，１２１が、優先順位比較モジュール１２４に対して各オペレーティングシステムの正規化優先順位を通知する。

【０１２０】

この実施例では、事務処理ＯＳ１１０，リアルタイムＯＳ１１１とオペレーティングシステム切替えプログラム１０８とのインタフェースが全て正規化優先順位によって行われる。前記第一乃至第三の実施例では、実行可能タスクが存在する場合には、タスクの優先順位を通知し、実行可能タスクが存在しない場合には、idle状態であることを通知していた。これらの場合、idle状態や、オペレーティングシステム自体が何らかの処理を実行しなければならない状態を、自由に正規化優先順位にマッピングすることができない。本実施例（第四実施例）では、各オペレーティングシステム内で優先順位を正規化優先順位に変換するため、タスク実行中以外の状態も自由に正規化優先順位にマッピングすることが可能となる。

【０１２１】

【発明の効果】

本発明によれば、複数個のオペレーティングシステムを単一プロセッサで動作させる計算機において、各オペレーティングシステムが実行するタスクの優先順位にしたがって、オペレーティングシステムの切り替えを行うことができ、より重要なタスクを優先して実行させることができる。また、本発明では、優先順位変換モジュールによって正規化優先順位に変換し、これによってオペレーティングシステム間の優先順位比較を行うため、異なった優先順位体系を有するオペレーティングシステム同士を実行させる計算機であっても、リアルタイム性・効率性を損なうことがない。

【図面の簡単な説明】

【図１】

本発明の第一の実施例におけるシステム構成を示す図である。

【図２】

計算機のハードウェア構成を示す図である。

【図３】

割込みレベルマスク機能を装備する場合のステータスレジスタを示す図である。

【図４】

個別割込みマスク機能を装備する場合のステータスレジスタを示す図である。

【図５】

オペレーティングシステム内部構成を詳細に示す図である。

【図６】

オペレーティングシステム切り替えプログラム内部構成を詳細に示す第一の図である。

【図７】

リスケジューラの処理フローを示す図である。

【図８】

優先順位変換モジュールの処理フローを示す図である。

【図９】

優先順位比較モジュールの処理フローを示す図である。

【図１０】

オペレーティングシステム切り替えプログラム内部構成を詳細に示す第二の図である。

【図１１】

ＯＳコンテクスト切り替えモジュールの処理フローを示す図である。

【図１２】

共通割込みハンドラの処理フローを示す図である。

【図１３】

割込みハンドラの処理フローを示す図である。

【図１４】

優先順位上昇方式におけるロック取得モジュールの処理フローを示す図である。

【図１５】

優先順位上昇方式におけるロック解放モジュールの処理フローを示す図である。

【図１６】

優先順位設定モジュール内部構成を詳細に示す図である。

【図１７】

優先順位変換モジュールにおける優先順位逆変換機能の処理フローを示す図である。

【図１８】

優先順位設定モジュールの処理フローを示す図である。

【図１９】

優先順位継承方式におけるロック取得モジュールの処理フローを示す図である。

【図２０】

優先順位継承方式におけるロック解放モジュールの処理フローを示す図である。

【図２１】

割込みマスクレベル計算モジュール内部構成を詳細に示す図である。

【図２２】

割込みマスクレベル計算モジュールの処理フローを示す図である。

【図２３】

本発明の第二の実施例におけるシステム構成を示す図である。

【図２４】

優先順位監視モジュールの処理フローを示す図である。

【図２５】

優先順位監視モジュールを起動する共通割込みハンドラの処理フローを示す図である。

【図２６】

本発明の第三の実施例におけるシステム構成を示す図である。

【図２７】

オペレーティングシステム間でのタスク振り分け例を示す図である。

【図２８】

共通割込みハンドラの別の構成を示した図である。

【図２９】

本発明の第四の実施例におけるシステム構成を示す図である。

【符号の説明】

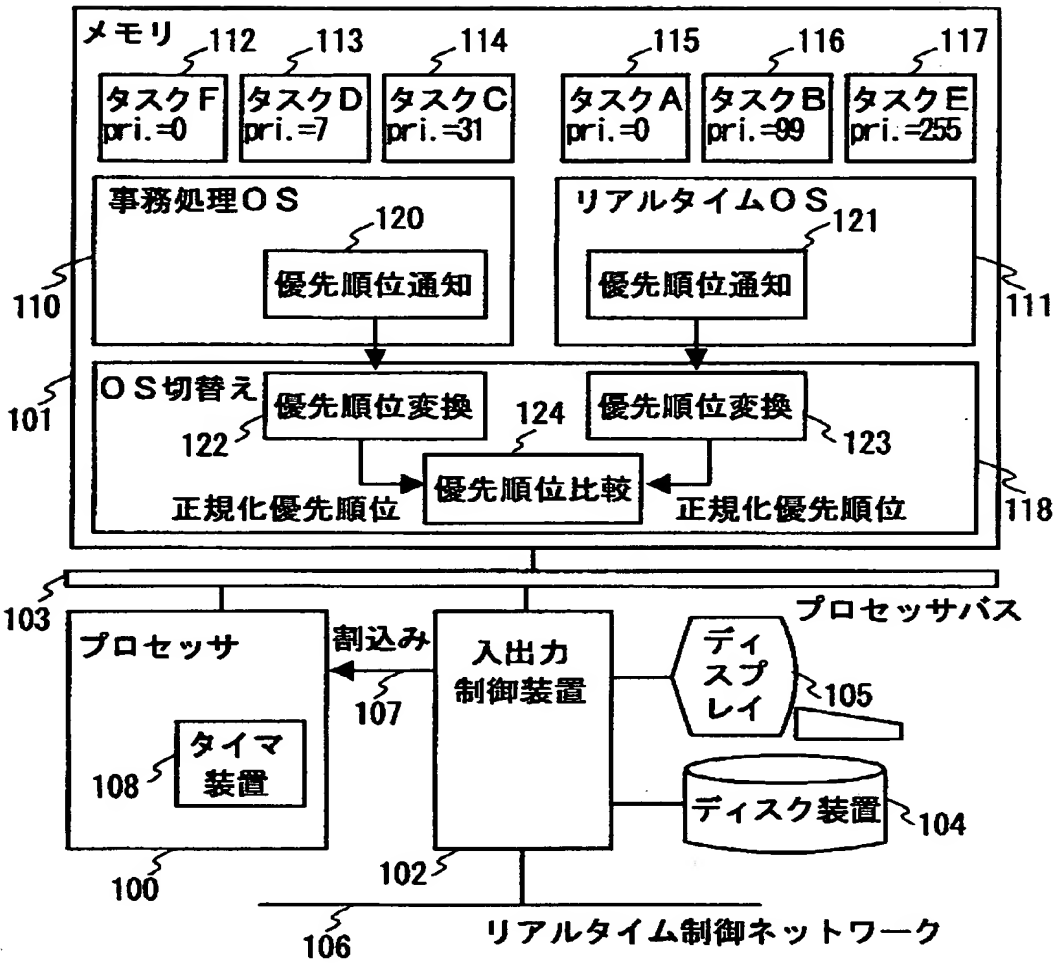
１００…プロセッサ、１０１…メモリ、１０２…入出力制御装置、１０３…プロセッサバス、１０４…ディスク装置、１０５…ディスプレイ、１０６…リアルタイム制御ネットワーク、１０７…割込み信号線、１０８…タイマ装置、１１０…事務処理ＯＳ、１１１…リアルタイムＯＳ、１１２～１１７，３７０～３７３…タスク、１１８…オペレーティングシステム切り替えプログラム、１２０，１２１…優先順位通知モジュール、１２２，１２３…優先順位変換モジュール、１２４…優先順位比較モジュール、１３０…ＣＰＵ、１３１…キャッシュメモリ、１３２…汎用レジスタ、１３３…プログラムカウンタ、１３４…ステータスレジスタ、１３５…割込みコントローラ、１３６…データバス、１３７…アドレスバス、１３８…割込み状態信号、１４０…割込みブロックビット、１４１…割込みマスクレベルフィールド、１４２…実行状態レジスタ、１４３…割込みマスクレジスタ、１４４～１４７…割込みマスクビット、１５０，１５１…実行可能待ち行列、１５２，１５３…割込みハンドラ、１５４，１５５…リスケジューラ、１５６，１５７…システムコールプログラム、１６０～１６５…タスク管理テーブル、１７０，１７１…優先順位変換テーブル、１７２…事務処理ＯＳ正規化優先順位、１７３…リアルタイムＯＳ正規化優先順位、１７４…共通割込みハンドラ、１７５…ＯＳ間通信機能モジュール、１７６…ＯＳコンテクスト切り替えモ

ジュール、１７７…割込みマスクレベル計算モジュール、２１０…実行ＯＳ記憶変数、２１１…割込み対応テーブル、２１２…事務処理ＯＳ用保存コンテクスト、２１３…リアルタイムＯＳ用保存コンテクスト、２１４，２１５…割込みスタック、２１６，２１７…割込みスタックポインタ、２１８…共有メモリ、２１９…ロック取得モジュール、２２０…ロック解放モジュール、２８０，２８１…優先順位設定モジュール、２８２，２８３…優先順位逆変換テーブル、３２０…割込みマスク変換テーブル、３４０，３４１…実行優先順位、３４２…事務処理ＯＳ実行優先順位記憶値、３４３…リアルタイムＯＳ実行優先順位記憶値、344…優先順位監視モジュール、３８０…割込み優先順位対応テーブル。

【書類名】　図面

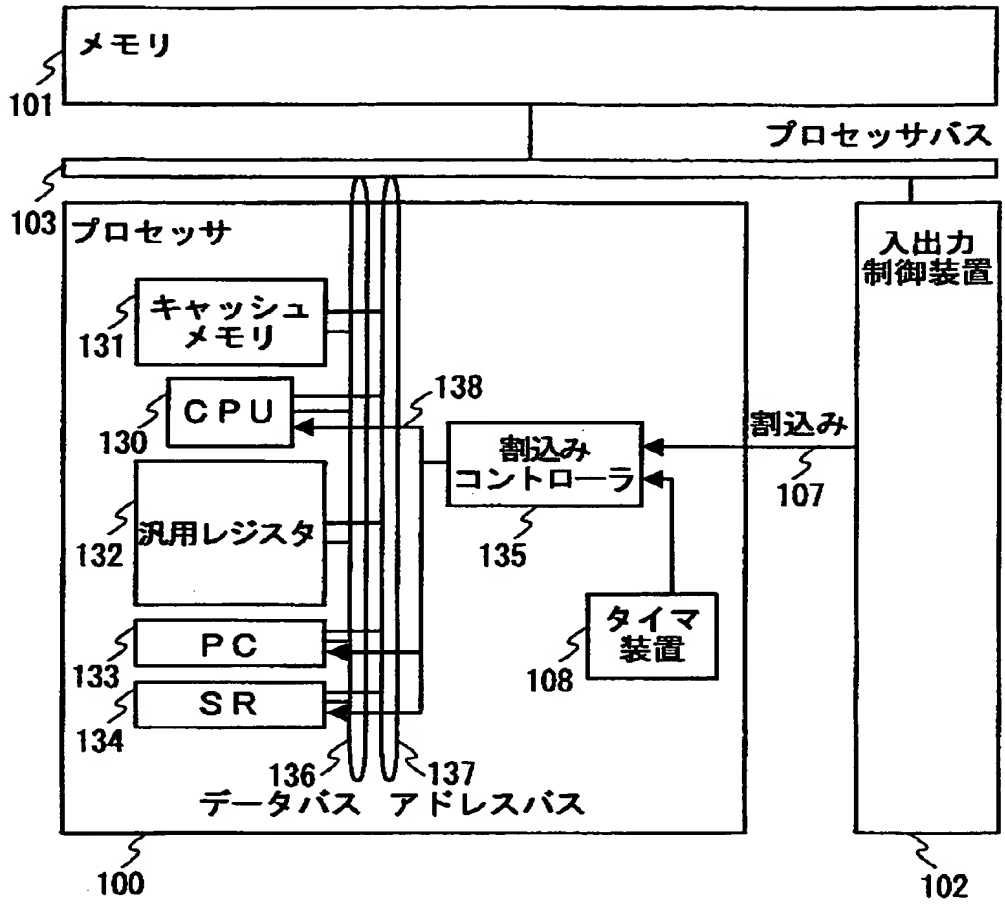
【図１】

図　1


メモリ
112
113
114
115
116
117
タスクF pri.=0
タスクD pri.=7
タスクC pri.=31
タスクA pri.=0
タスクB pri.=99
タスクE pri.=255
事務処理ＯＳ
120
優先順位通知
リアルタイムＯＳ
121
優先順位通知
110
111
ＯＳ切替え
101
優先順位変換
122
124
優先順位変換
123
優先順位比較
正規化優先順位
正規化優先順位
118
103
プロセッサバス
プロセッサ
割込み
107
入出力制御装置
ディスプレイ
105
タイマ装置
108
ディスク装置
104
100
102
106
リアルタイム制御ネットワーク

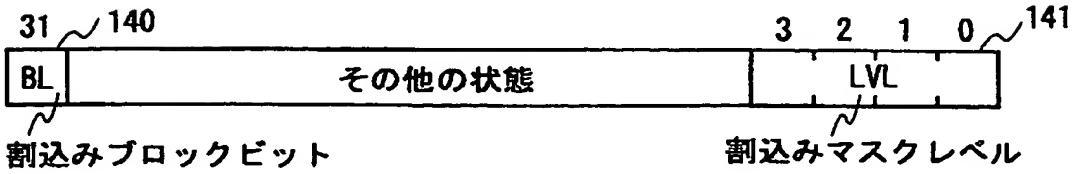
【図２】

図　2


メモリ
101
プロセッサバス
103
プロセッサ
入出力
制御装置
キャッシュ
メモリ
131
CPU
130
138
割込み
コントローラ
135
割込み
107
汎用レジスタ
132
タイマ
装置
108
PC
133
SR
134
136
137
データバス
アドレスバス
100
102

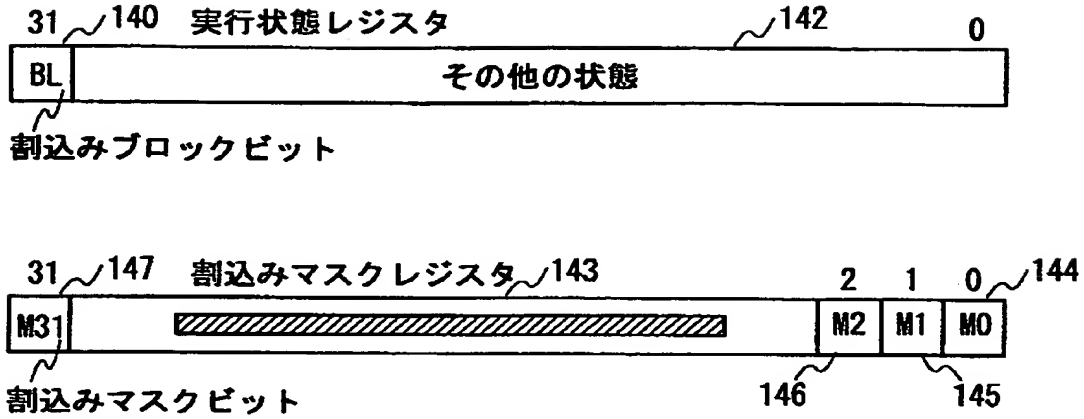
【図３】

図　3


31
140
3
2
1
0
141
BL
その他の状態
LVL
割込みブロックビット
割込みマスクレベル

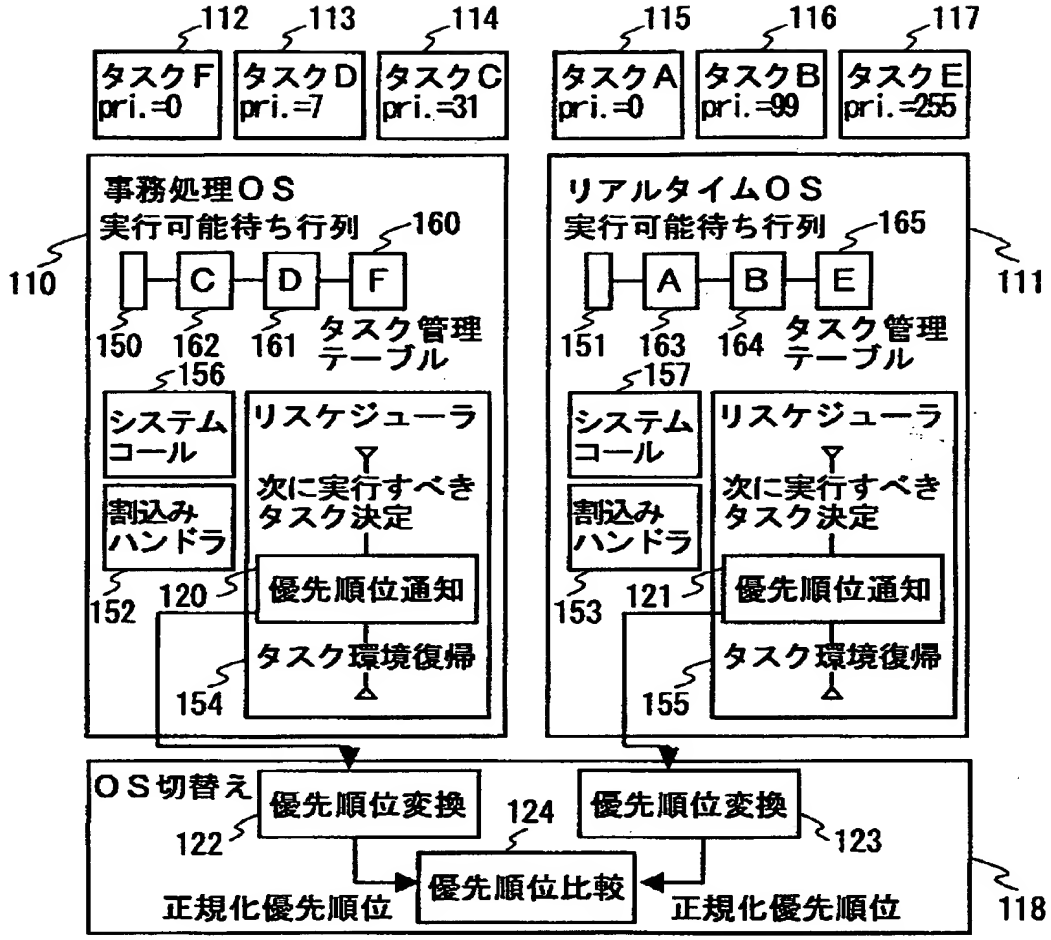
【図４】

図　4


31
140
実行状態レジスタ
142
0
BL
その他の状態
割込みブロックビット
31
147
割込みマスクレジスタ
143
2
1
0
144
M31
M2
M1
M0
割込みマスクビット
146
145

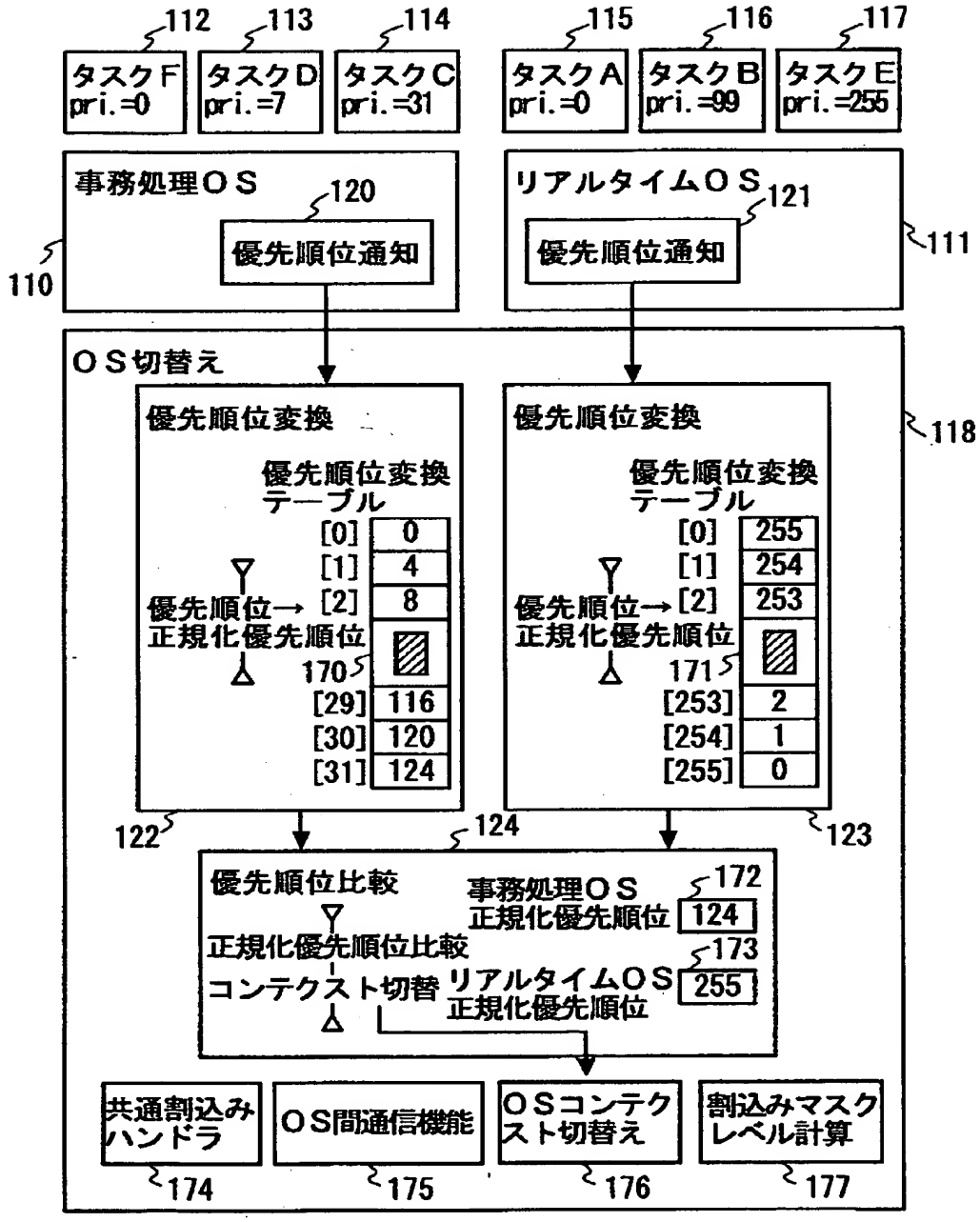
【図５】

図　5


112
113
114
115
116
117
タスクF pri.=0
タスクD pri.=7
タスクC pri.=31
タスクA pri.=0
タスクB pri.=99
タスクE pri.=255
事務処理ＯＳ
実行可能待ち行列
160
110
C
D
F
150
162
161
タスク管理テーブル
156
システムコール
割込みハンドラ
152
120
リスケジューラ
次に実行すべきタスク決定
優先順位通知
タスク環境復帰
154
リアルタイムＯＳ
実行可能待ち行列
165
111
A
B
E
151
163
164
タスク管理テーブル
157
システムコール
割込みハンドラ
153
121
リスケジューラ
次に実行すべきタスク決定
優先順位通知
タスク環境復帰
155
ＯＳ切替え
優先順位変換
122
124
優先順位変換
123
優先順位比較
正規化優先順位
正規化優先順位
118

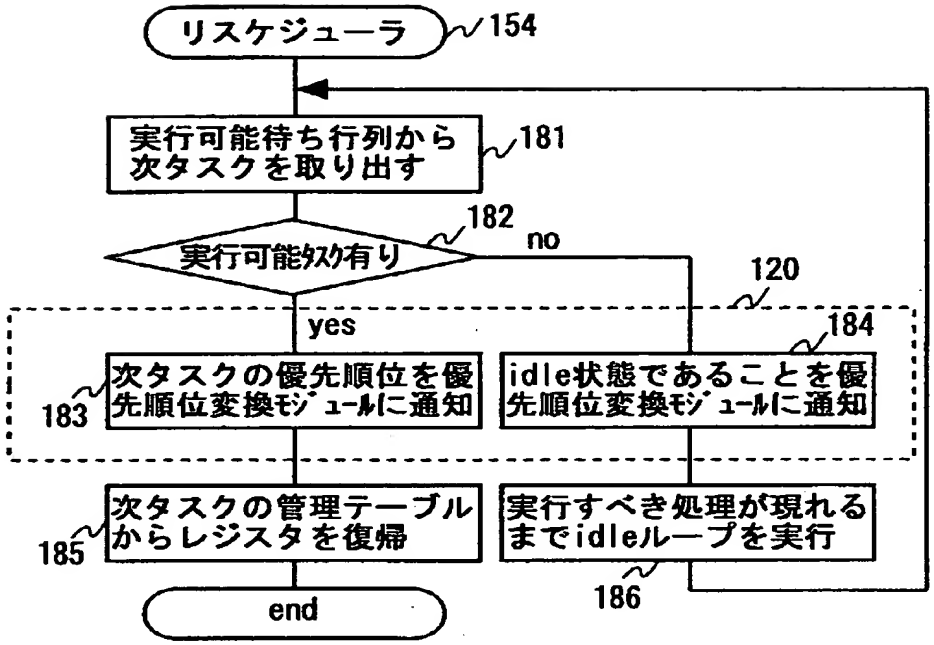
【図６】

図　6


112
113
114
115
116
117
タスクF
pri.=0
タスクD
pri.=7
タスクC
pri.=31
タスクA
pri.=0
タスクB
pri.=99
タスクE
pri.=255
事務処理ＯＳ
120
優先順位通知
110
リアルタイムＯＳ
121
優先順位通知
111
ＯＳ切替え
118
優先順位変換
優先順位変換テーブル
[0] 0
[1] 4
優先順位→ [2] 8
正規化優先順位
170
[29] 116
[30] 120
[31] 124
122
優先順位変換
優先順位変換テーブル
[0] 255
[1] 254
優先順位→ [2] 253
正規化優先順位
171
[253] 2
[254] 1
[255] 0
123
124
優先順位比較
正規化優先順位比較
コンテクスト切替
事務処理ＯＳ
正規化優先順位
172
124
リアルタイムＯＳ
正規化優先順位
173
255
共通割込みハンドラ
174
ＯＳ間通信機能
175
ＯＳコンテクスト切替え
176
割込みマスクレベル計算
177

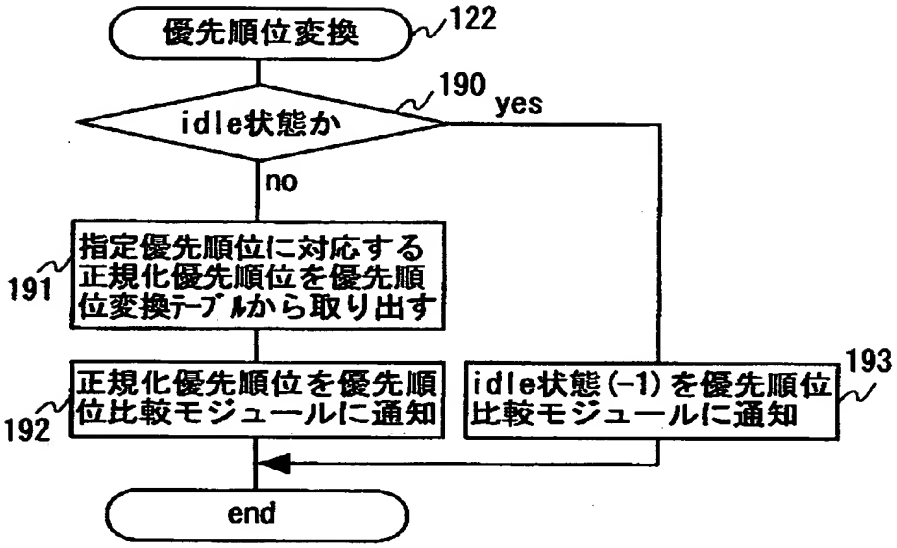
【図７】

図　7


リスケジューラ
154
実行可能待ち行列から
次タスクを取り出す
181
182
実行可能ﾀｽｸ有り
no
120
yes
184
183
次タスクの優先順位を優
先順位変換ﾓｼﾞｭｰﾙに通知
idle状態であることを優
先順位変換ﾓｼﾞｭｰﾙに通知
次タスクの管理テーブル
からレジスタを復帰
185
実行すべき処理が現れる
までidleループを実行
186
end

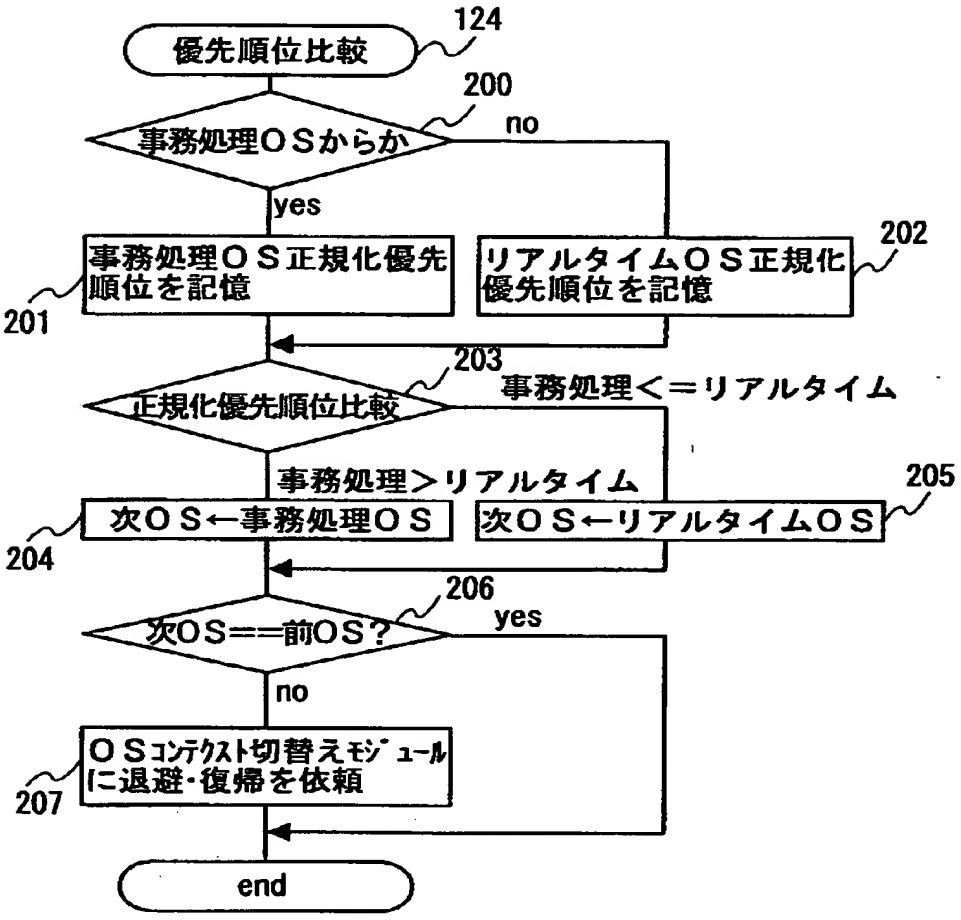
【図８】

図　8


優先順位変換
122
190
idle状態か
yes
no
191
指定優先順位に対応する正規化優先順位を優先順位変換テーブルから取り出す
192
正規化優先順位を優先順位比較モジュールに通知
193
idle状態(-1)を優先順位比較モジュールに通知
end

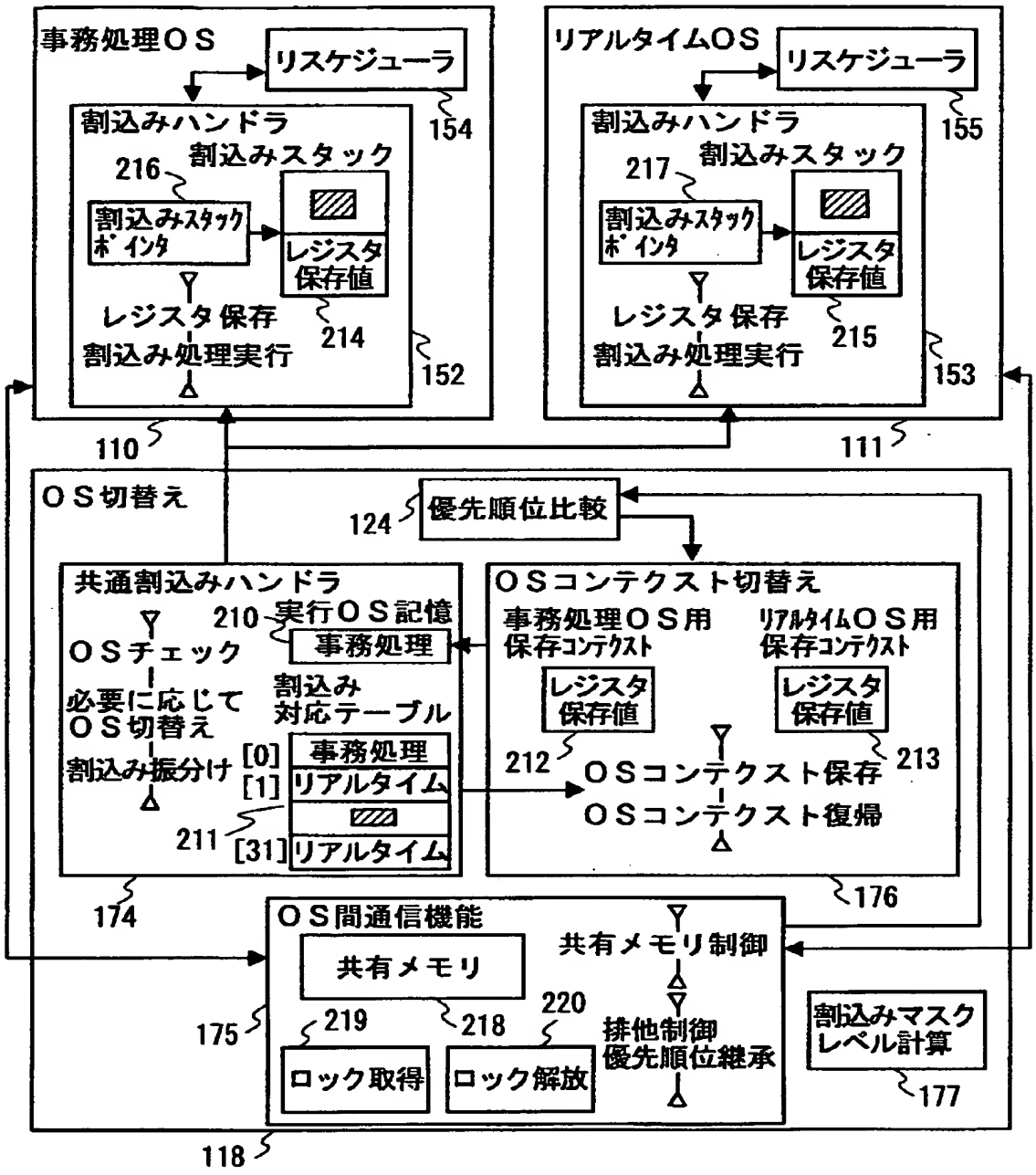
【図９】

図　9


124
優先順位比較
200
事務処理ＯＳからか
no
yes
事務処理ＯＳ正規化優先順位を記憶
201
リアルタイムＯＳ正規化優先順位を記憶
202
203
正規化優先順位比較
事務処理＜＝リアルタイム
事務処理＞リアルタイム
205
次ＯＳ←事務処理ＯＳ
204
次ＯＳ←リアルタイムＯＳ
206
次ＯＳ==前ＯＳ?
yes
no
ＯＳｺﾝﾃｸｽﾄ切替えﾓｼﾞｭｰﾙに退避･復帰を依頼
207
end

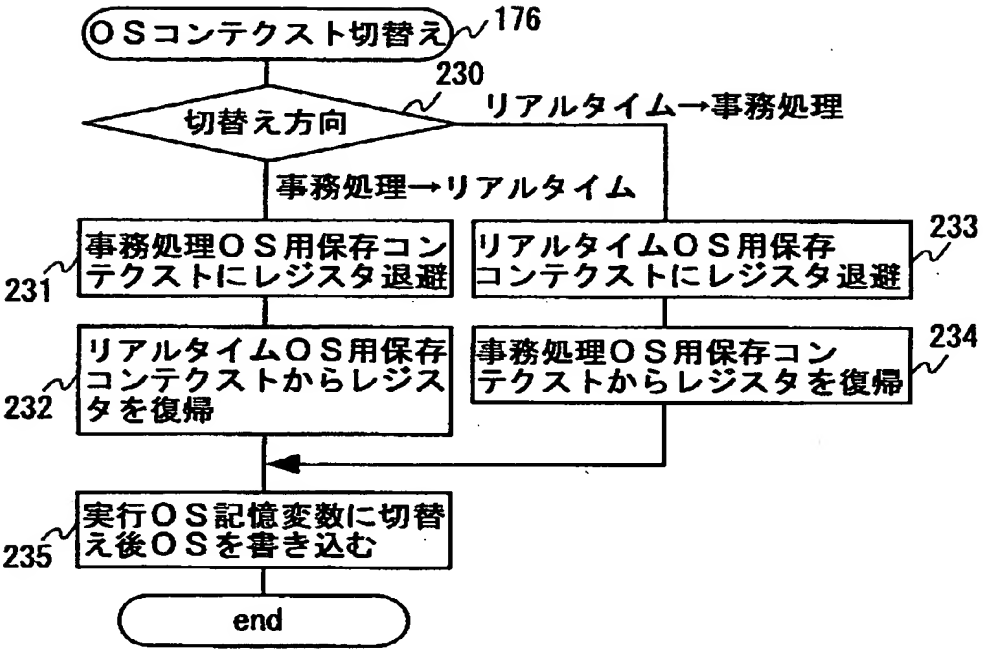
【図１０】

図　10


事務処理ＯＳ
リスケジューラ
154
割込みハンドラ
割込みスタック
216
割込みスタック
ポインタ
レジスタ
保存値
214
レジスタ保存
割込み処理実行
152
110
リアルタイムOS
リスケジューラ
155
割込みハンドラ
割込みスタック
217
割込みスタック
ポインタ
レジスタ
保存値
215
レジスタ保存
割込み処理実行
153
111
ＯＳ切替え
124
優先順位比較
共通割込みハンドラ
210
実行ＯＳ記憶
事務処理
ＯＳチェック
必要に応じて
ＯＳ切替え
割込み
対応テーブル
割込み振分け
[0]
事務処理
[1]
リアルタイム
211
[31]
リアルタイム
174
ＯＳコンテクスト切替え
事務処理ＯＳ用
保存コンテクスト
リアルタイムＯＳ用
保存コンテクスト
レジスタ
保存値
レジスタ
保存値
212
213
ＯＳコンテクスト保存
ＯＳコンテクスト復帰
176
ＯＳ間通信機能
共有メモリ制御
共有メモリ
220
175
219
218
排他制御
優先順位継承
ロック取得
ロック解放
割込みマスク
レベル計算
177
118

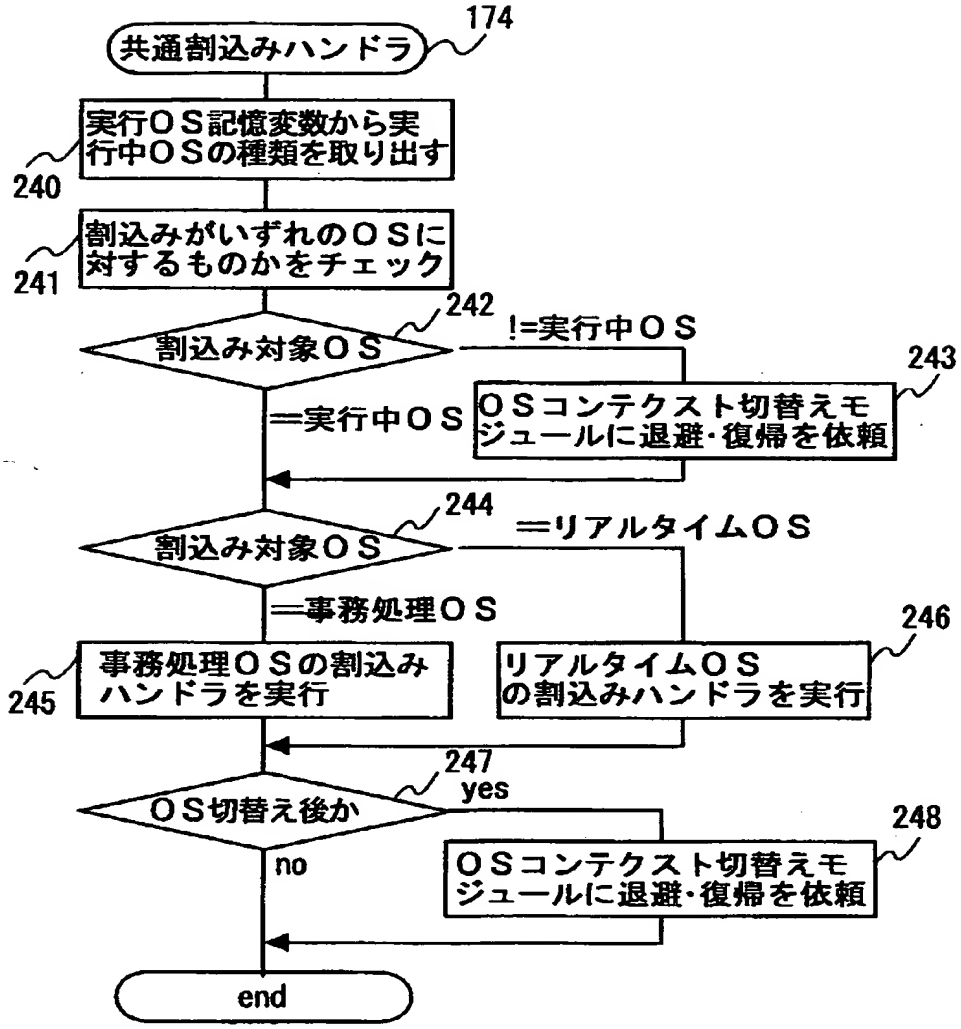
【図１１】

図 11


ＯＳコンテクスト切替え
176
230
切替え方向
リアルタイム→事務処理
事務処理→リアルタイム
事務処理ＯＳ用保存コンテクストにレジスタ退避
231
リアルタイムＯＳ用保存コンテクストにレジスタ退避
233
リアルタイムＯＳ用保存コンテクストからレジスタを復帰
232
事務処理ＯＳ用保存コンテクストからレジスタを復帰
234
実行ＯＳ記憶変数に切替え後ＯＳを書き込む
235
end

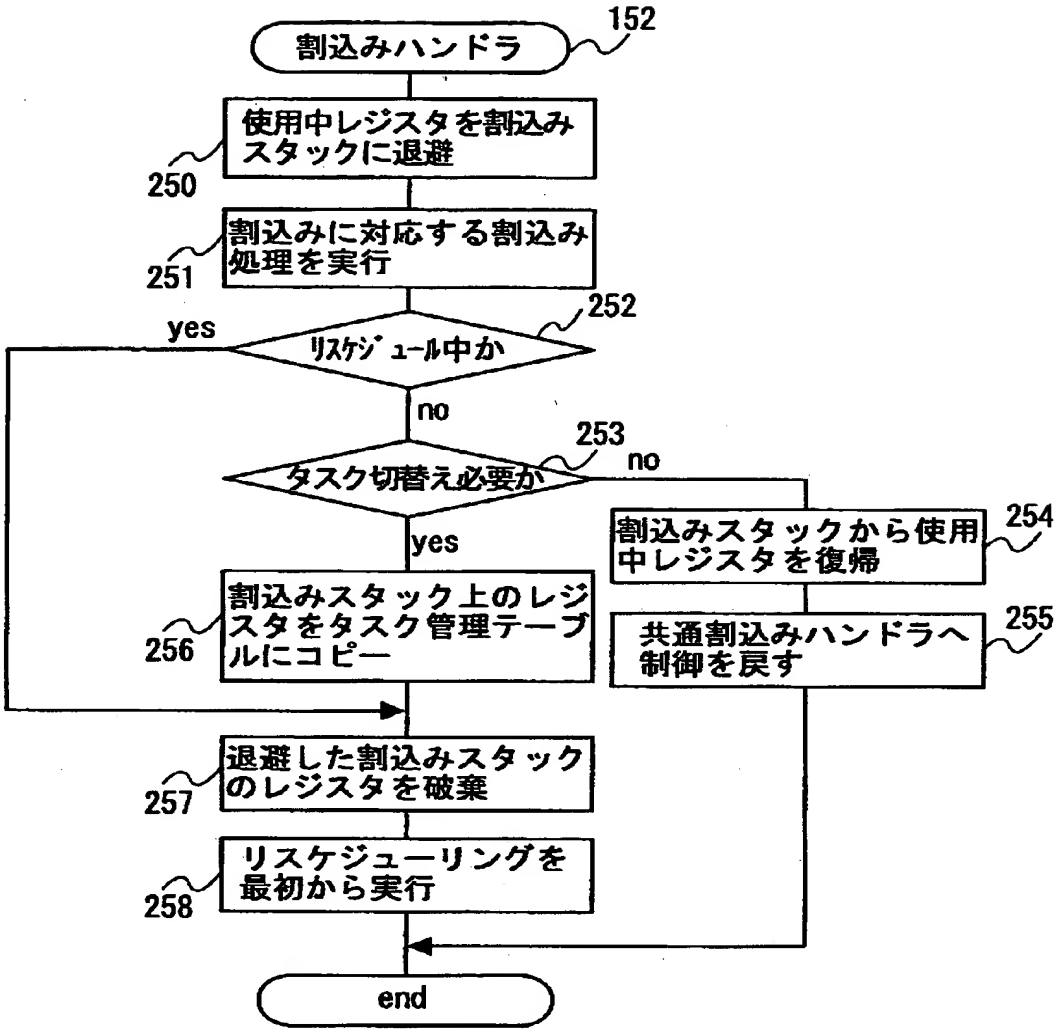
【図１２】

図　12


共通割込みハンドラ
174
実行ＯＳ記憶変数から実行中ＯＳの種類を取り出す
240
割込みがいずれのＯＳに対するものかをチェック
241
割込み対象ＯＳ
242
!=実行中ＯＳ
243
ＯＳコンテクスト切替えモジュールに退避･復帰を依頼
==実行中ＯＳ
割込み対象ＯＳ
244
==リアルタイムＯＳ
==事務処理ＯＳ
事務処理ＯＳの割込みハンドラを実行
245
リアルタイムＯＳの割込みハンドラを実行
246
ＯＳ切替え後か
247
yes
no
ＯＳコンテクスト切替えモジュールに退避･復帰を依頼
248
end

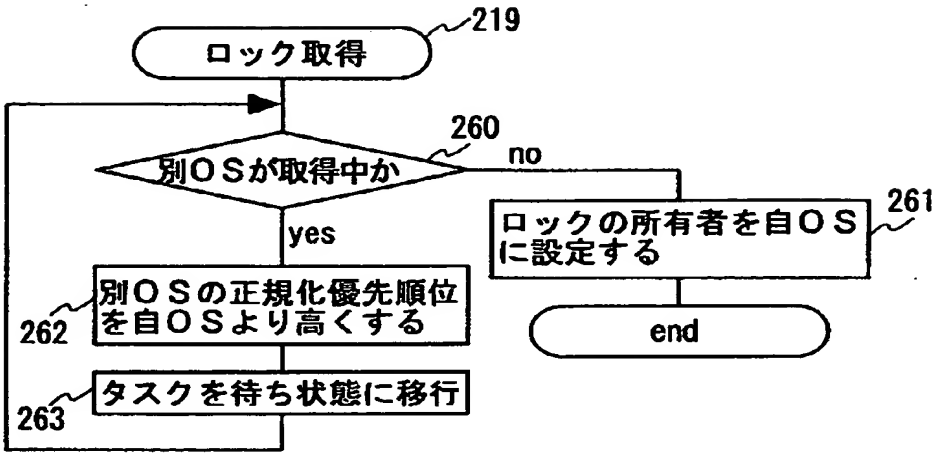
【図１３】

図　13


割込みハンドラ
152
使用中レジスタを割込みスタックに退避
250
割込みに対応する割込み処理を実行
251
リスケジュール中か
252
yes
no
タスク切替え必要か
253
no
yes
割込みスタックから使用中レジスタを復帰
254
割込みスタック上のレジスタをタスク管理テーブルにコピー
256
共通割込みハンドラへ制御を戻す
255
退避した割込みスタックのレジスタを破棄
257
リスケジューリングを最初から実行
258
end

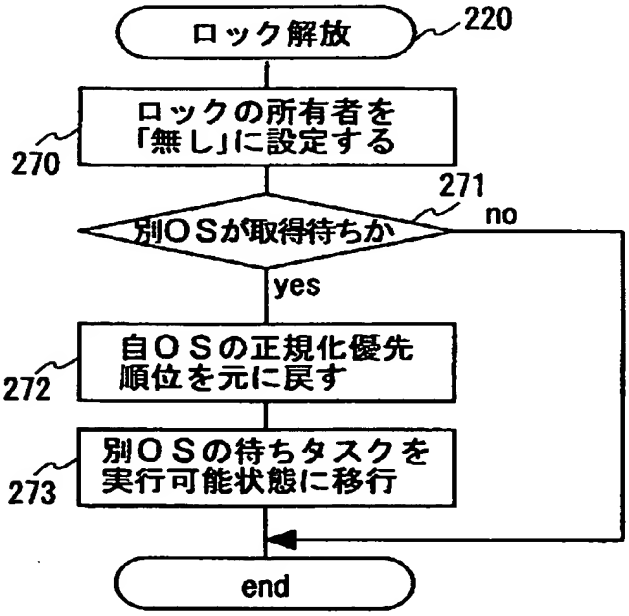
【図１４】

図　14



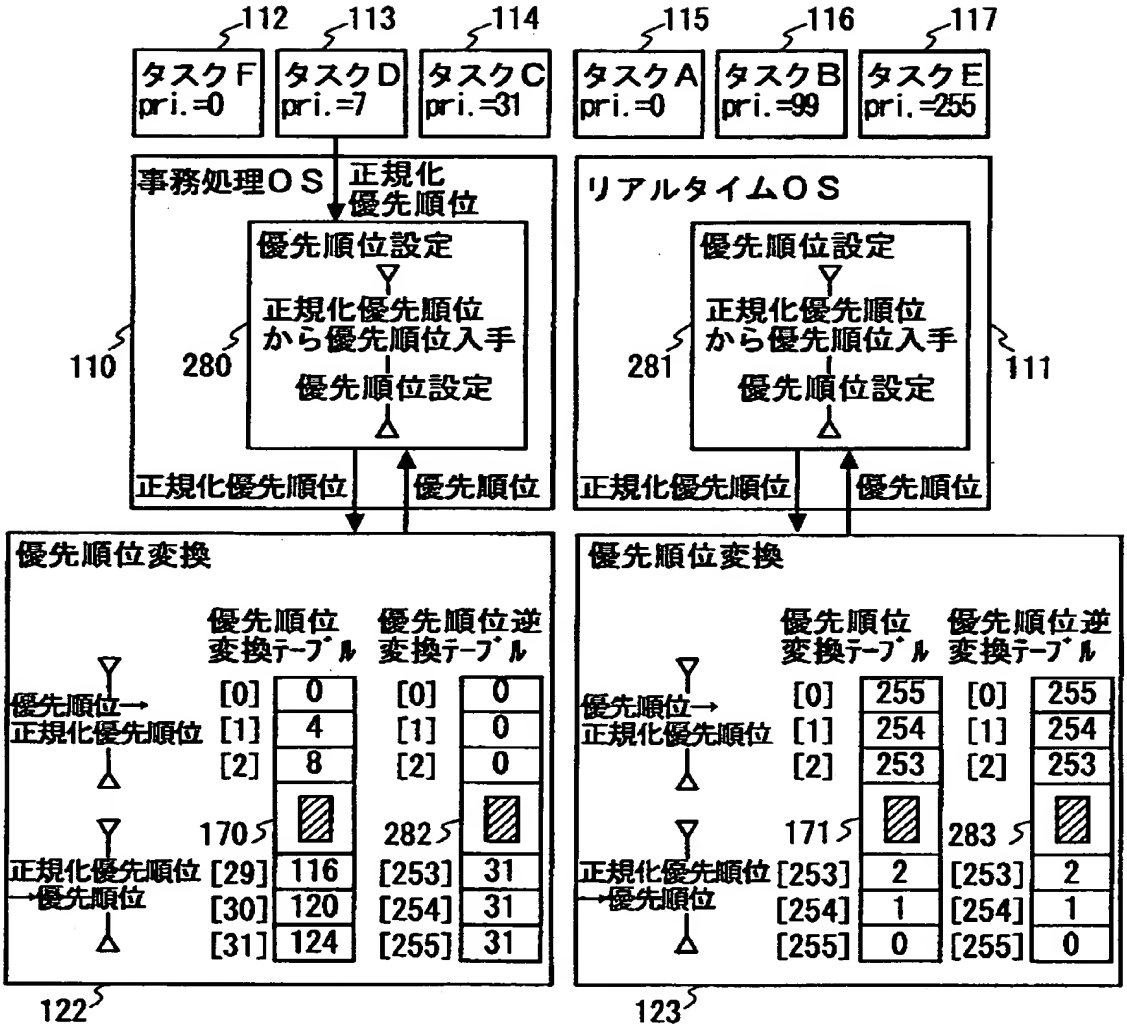
【図１５】

図　15



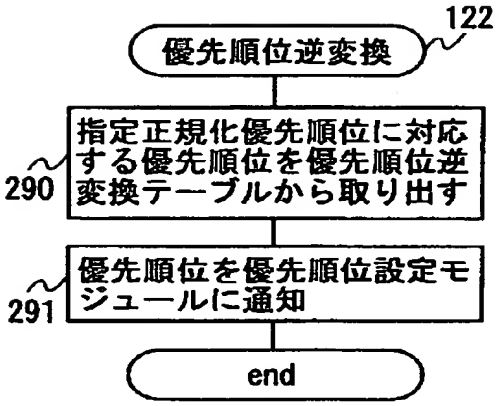
【図１６】

図　16


112
113
114
115
116
117
タスクF
pri.=0
タスクD
pri.=7
タスクC
pri.=31
タスクA
pri.=0
タスクB
pri.=99
タスクE
pri.=255
事務処理ＯＳ
正規化
優先順位
リアルタイムＯＳ
優先順位設定
正規化優先順位
から優先順位入手
優先順位設定
110
280
281
111
正規化優先順位
優先順位
優先順位変換
優先順位
変換ﾃｰﾌﾞﾙ
優先順位逆
変換ﾃｰﾌﾞﾙ
優先順位→
正規化優先順位
正規化優先順位
→優先順位
[0] 0
[1] 4
[2] 8
170
[29] 116
[30] 120
[31] 124
[0] 0
[1] 0
[2] 0
282
[253] 31
[254] 31
[255] 31
[0] 255
[1] 254
[2] 253
171
[253] 2
[254] 1
[255] 0
[0] 255
[1] 254
[2] 253
283
[253] 2
[254] 1
[255] 0
122
123

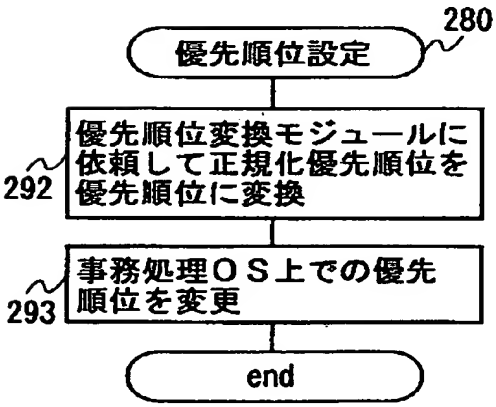
【図１７】

図　17



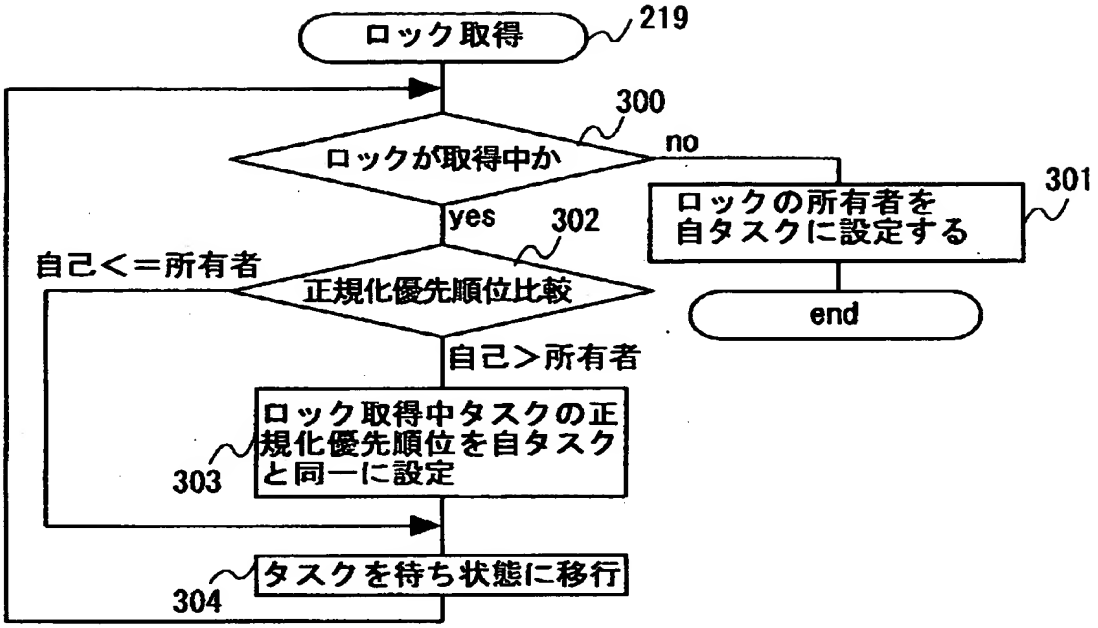
【図１８】

図　18



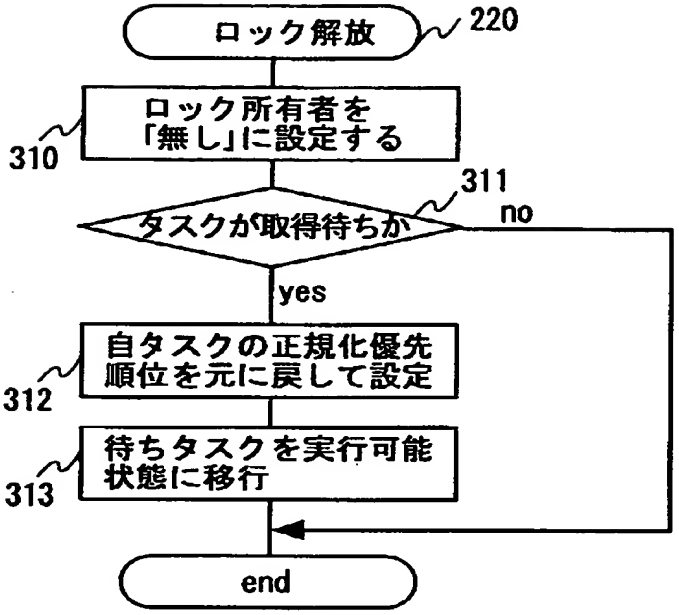
【図１９】

図 19


ロック取得
219
300
ロックが取得中か
no
301
ロックの所有者を
自タスクに設定する
yes
302
自己<＝所有者
正規化優先順位比較
end
自己＞所有者
ロック取得中タスクの正
規化優先順位を自タスク
と同一に設定
303
タスクを待ち状態に移行
304

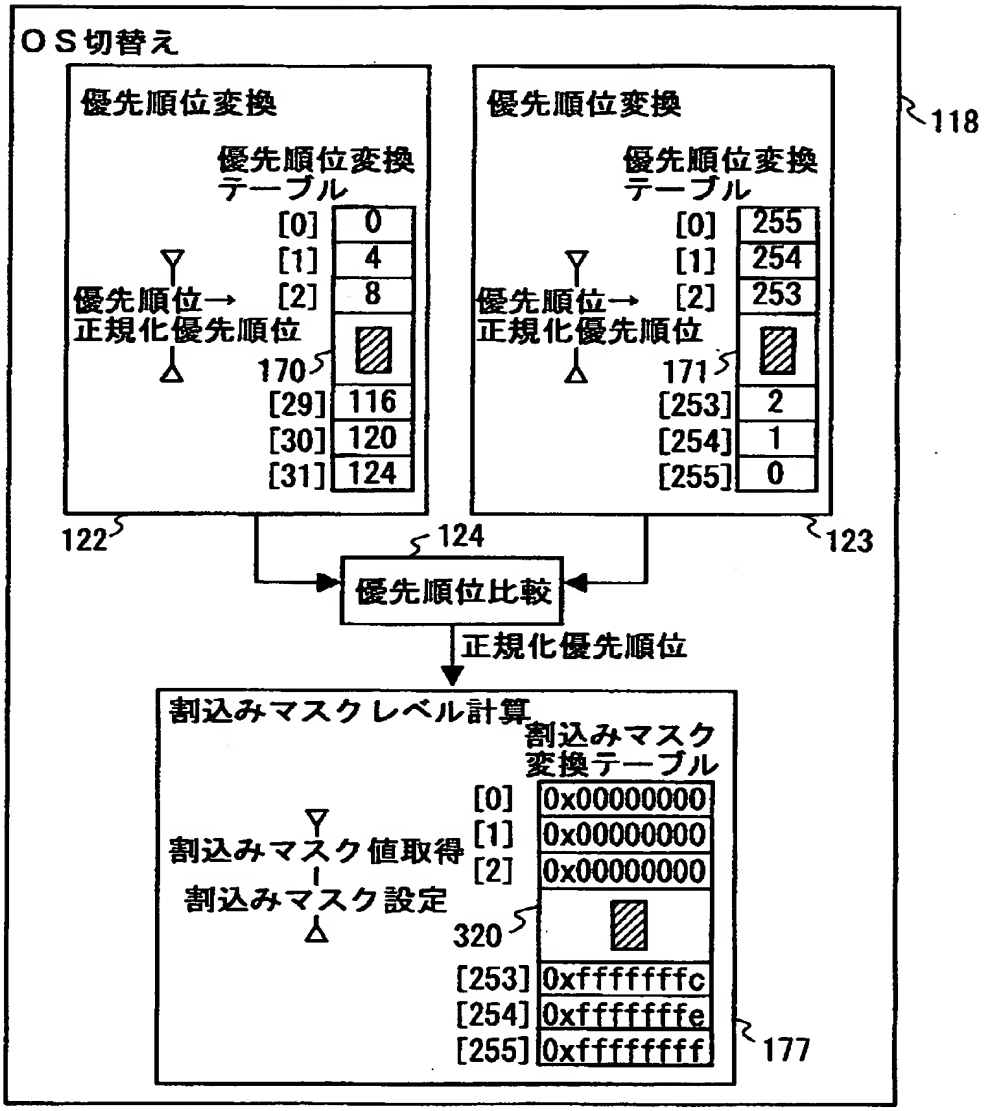
【図２０】

図　20


ロック解放
220
ロック所有者を
「無し」に設定する
310
311
タスクが取得待ちか
no
yes
自タスクの正規化優先
順位を元に戻して設定
312
待ちタスクを実行可能
状態に移行
313
end

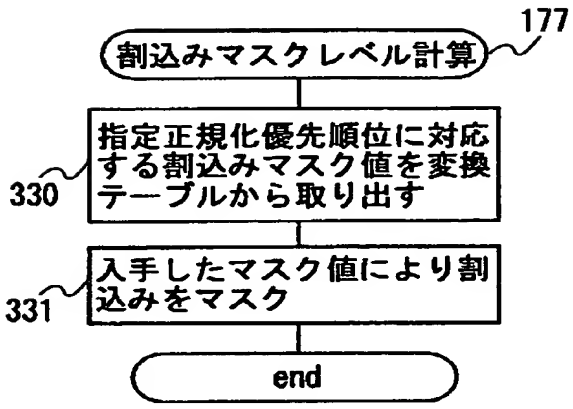
【図２１】

図　21


ＯＳ切替え
優先順位変換
優先順位変換テーブル
[0] 0
[1] 4
[2] 8
優先順位→正規化優先順位
170
[29] 116
[30] 120
[31] 124
122
優先順位変換
優先順位変換テーブル
[0] 255
[1] 254
[2] 253
優先順位→正規化優先順位
171
[253] 2
[254] 1
[255] 0
123
118
124
優先順位比較
正規化優先順位
割込みマスクレベル計算
割込みマスク変換テーブル
[0] 0x00000000
[1] 0x00000000
[2] 0x00000000
割込みマスク値取得
割込みマスク設定
320
[253] 0xfffffffc
[254] 0xfffffffe
[255] 0xffffffff
177

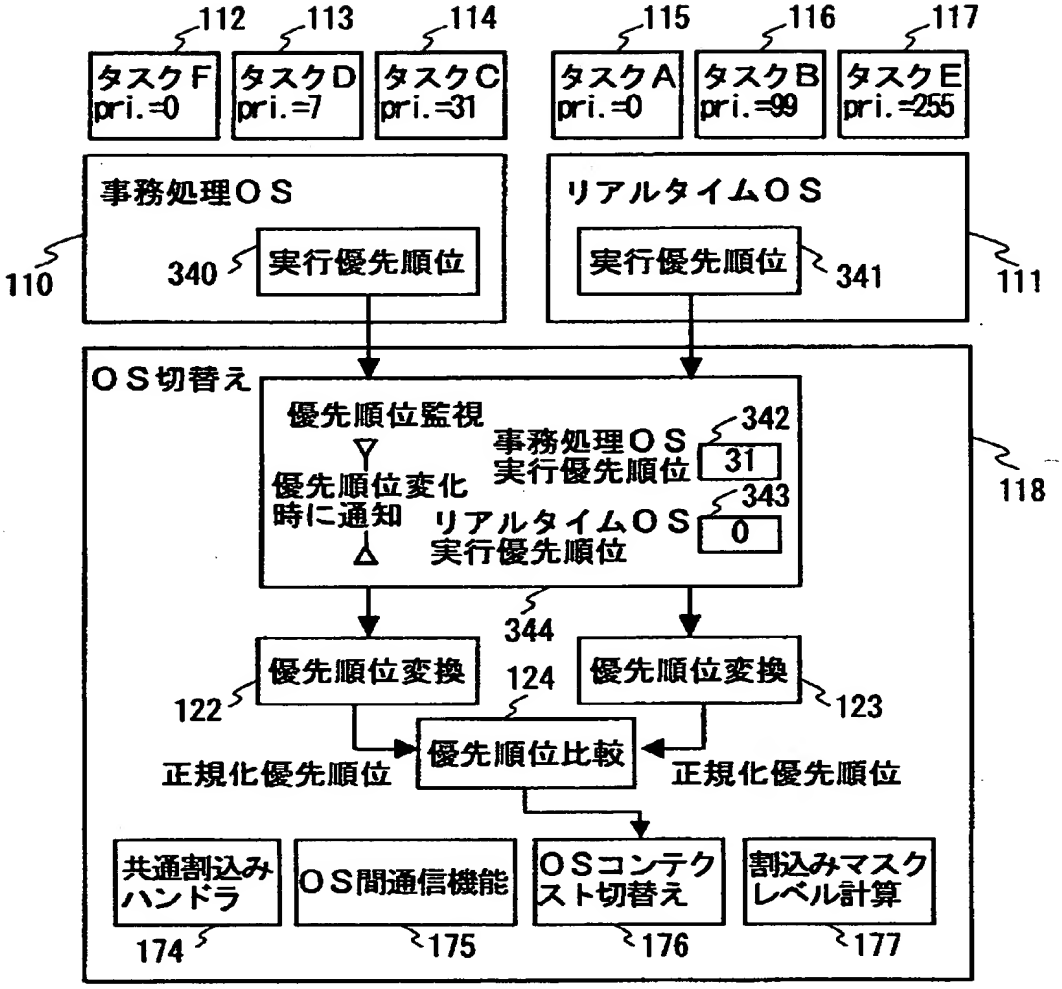
【図２２】

図　22


177
割込みマスクレベル計算
330
指定正規化優先順位に対応する割込みマスク値を変換テーブルから取り出す
331
入手したマスク値により割込みをマスク
end

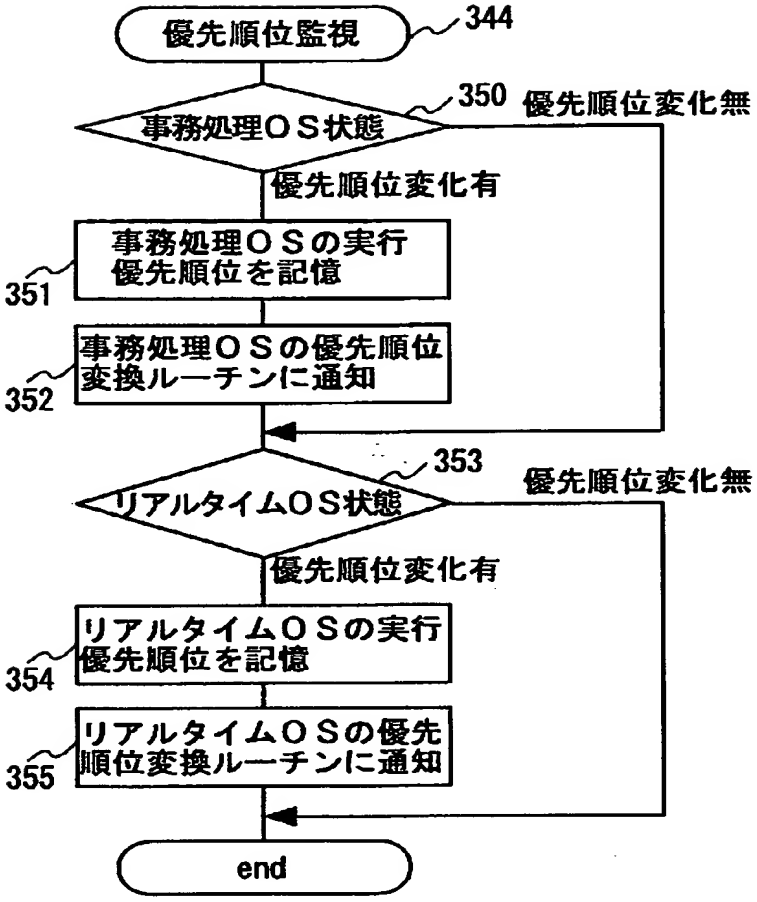
【図２３】

図　23


112
113
114
115
116
117
タスクF pri.=0
タスクD pri.=7
タスクC pri.=31
タスクA pri.=0
タスクB pri.=99
タスクE pri.=255
事務処理ＯＳ
リアルタイムＯＳ
110
340
実行優先順位
実行優先順位
341
111
ＯＳ切替え
優先順位監視
優先順位変化時に通知
事務処理ＯＳ実行優先順位
342
31
リアルタイムＯＳ実行優先順位
343
0
118
344
優先順位変換
優先順位変換
122
124
123
優先順位比較
正規化優先順位
正規化優先順位
共通割込みハンドラ
ＯＳ間通信機能
ＯＳコンテクスト切替え
割込みマスクレベル計算
174
175
176
177

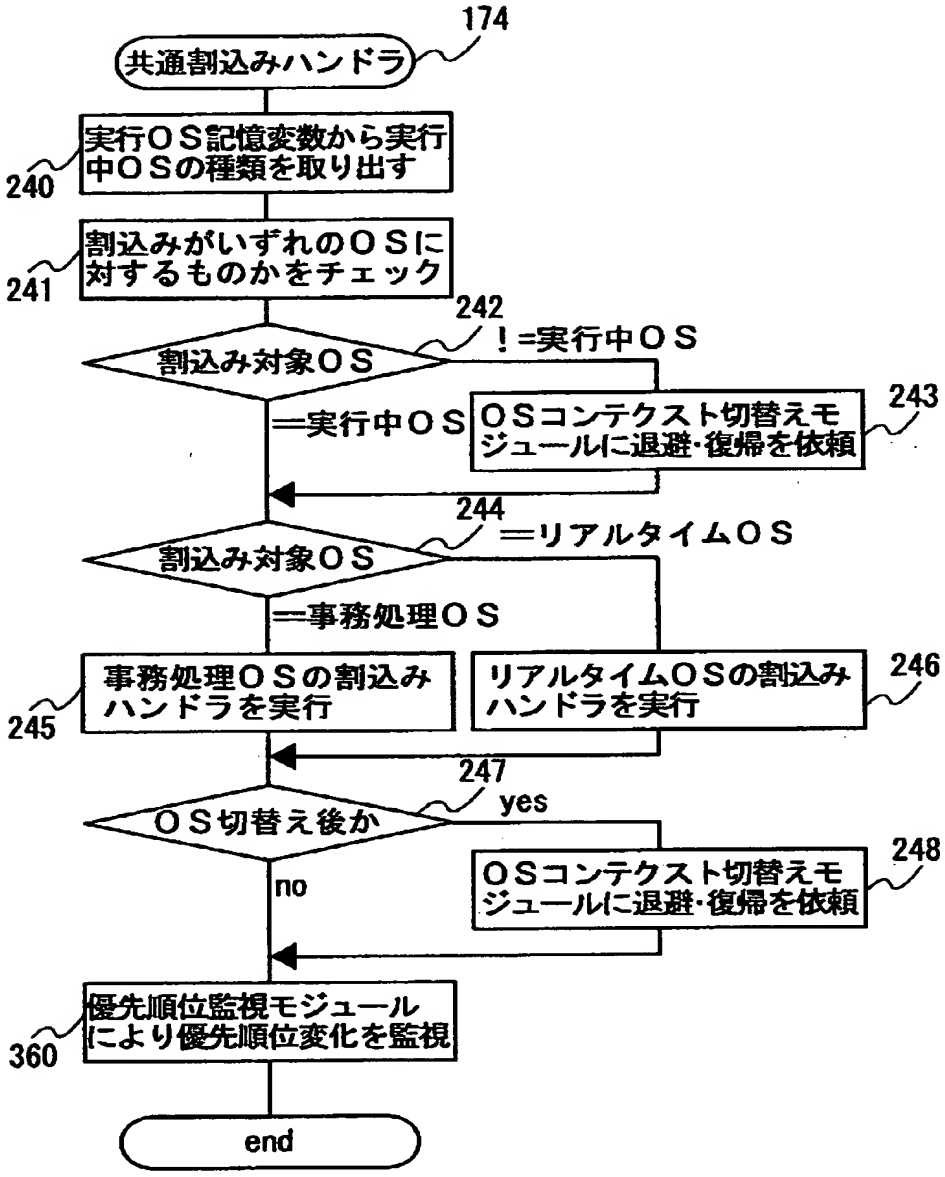
【図２４】

図　24


優先順位監視
344
事務処理ＯＳ状態
350
優先順位変化無
優先順位変化有
事務処理ＯＳの実行
優先順位を記憶
351
事務処理ＯＳの優先順位
変換ルーチンに通知
352
リアルタイムＯＳ状態
353
優先順位変化無
優先順位変化有
リアルタイムＯＳの実行
優先順位を記憶
354
リアルタイムＯＳの優先
順位変換ルーチンに通知
355
end

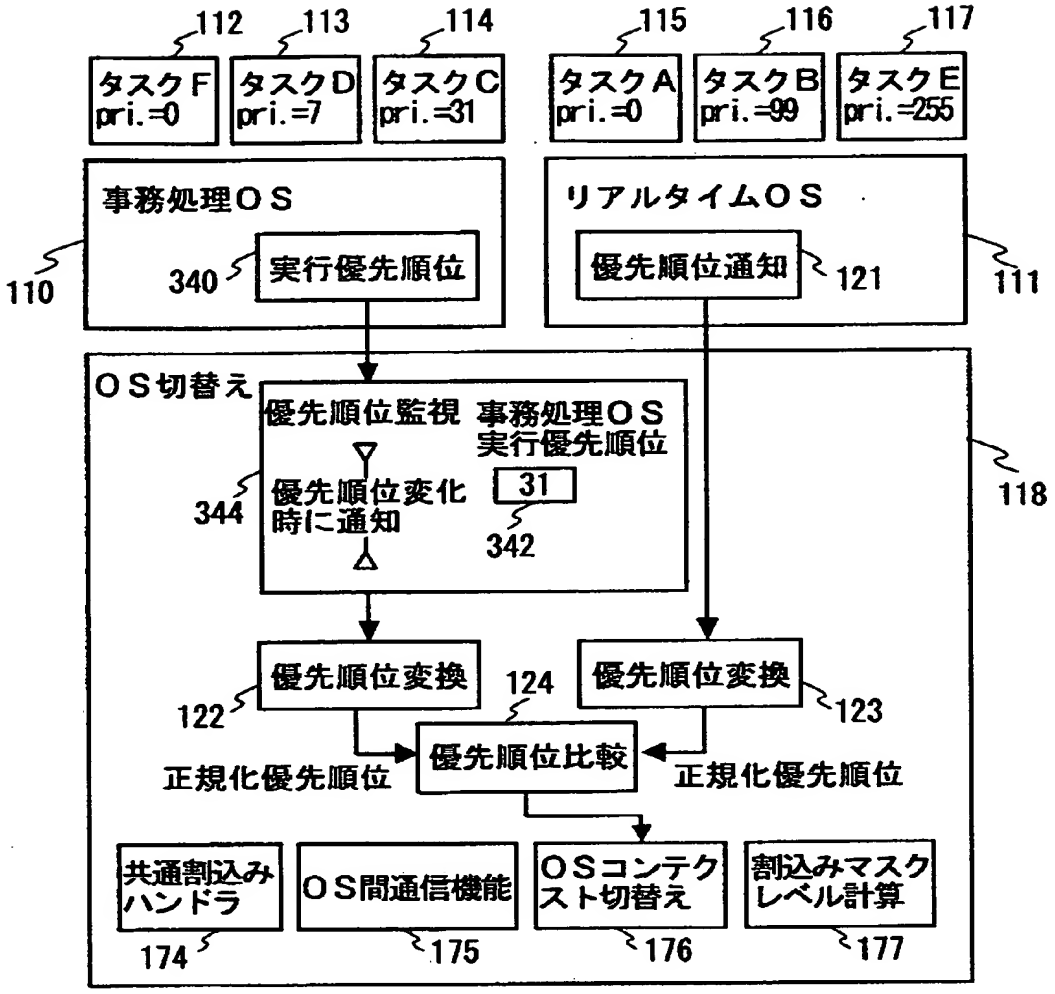
【図２５】

図　25


174
共通割込みハンドラ
実行ＯＳ記憶変数から実行中ＯＳの種類を取り出す
240
割込みがいずれのＯＳに対するものかをチェック
241
242
割込み対象ＯＳ
！＝実行中ＯＳ
＝実行中ＯＳ
ＯＳコンテクスト切替えモジュールに退避・復帰を依頼
243
244
割込み対象ＯＳ
＝リアルタイムＯＳ
＝事務処理ＯＳ
事務処理ＯＳの割込みハンドラを実行
245
リアルタイムＯＳの割込みハンドラを実行
246
247
ＯＳ切替え後か
yes
no
ＯＳコンテクスト切替えモジュールに退避・復帰を依頼
248
優先順位監視モジュールにより優先順位変化を監視
360
end

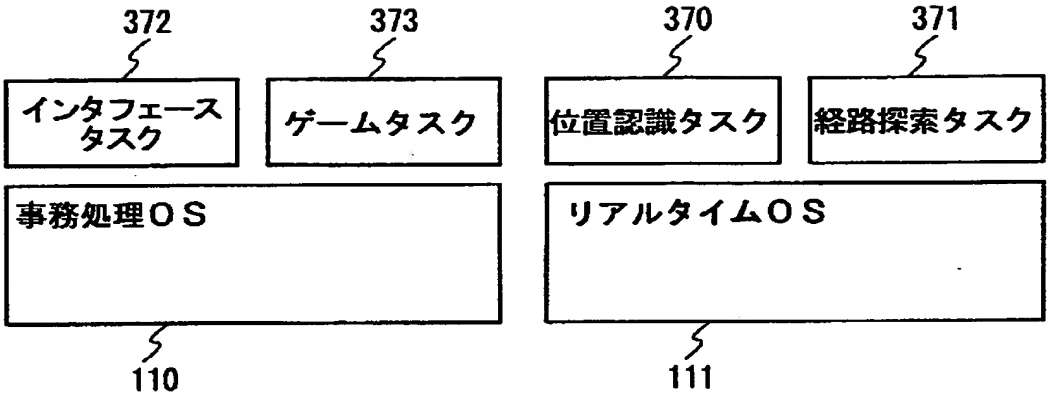
【図２６】

図　26


112
113
114
115
116
117
タスクF pri.=0
タスクD pri.=7
タスクC pri.=31
タスクA pri.=0
タスクB pri.=99
タスクE pri.=255
事務処理ＯＳ
リアルタイムＯＳ
110
340
実行優先順位
優先順位通知
121
111
ＯＳ切替え
優先順位監視
事務処理ＯＳ
実行優先順位
優先順位変化
時に通知
344
31
342
118
優先順位変換
優先順位変換
122
124
123
優先順位比較
正規化優先順位
正規化優先順位
共通割込み
ハンドラ
ＯＳ間通信機能
ＯＳコンテク
スト切替え
割込みマスク
レベル計算
174
175
176
177

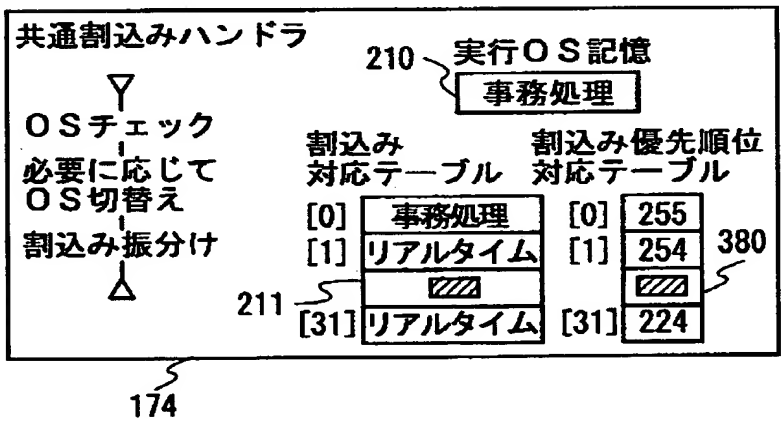
【図２７】

図　27



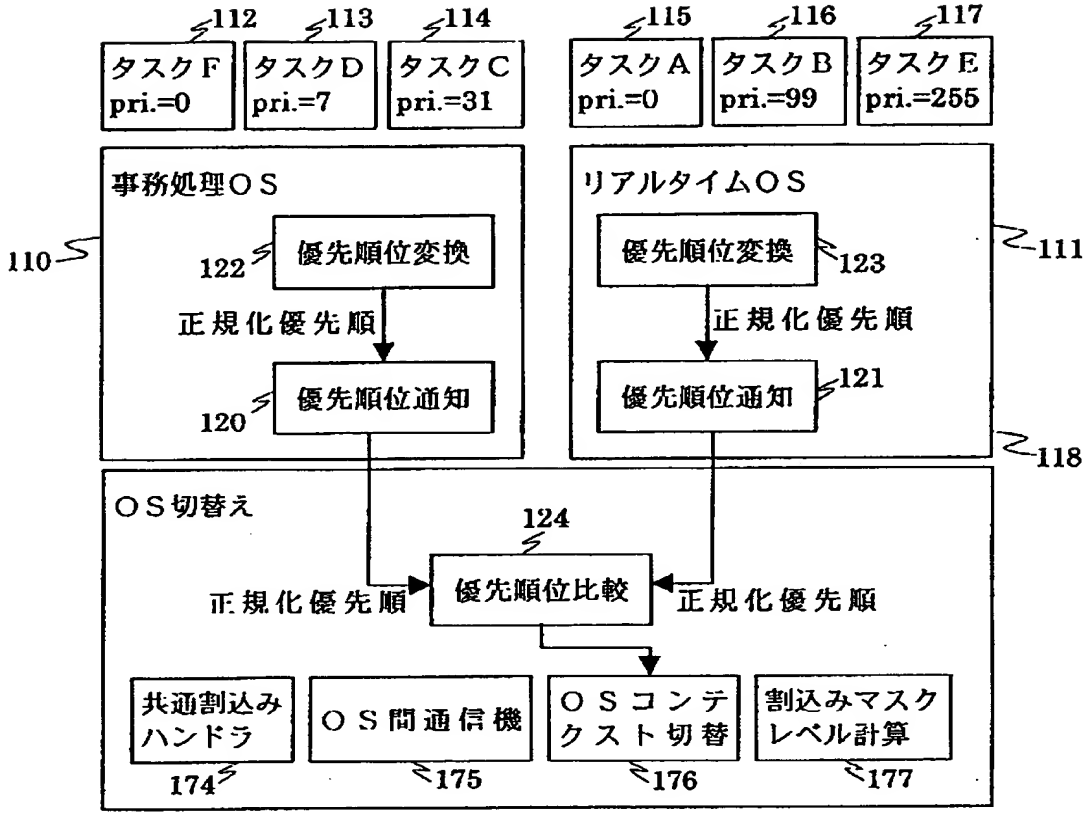
【図２８】

図　28



【図２９】

図　29


112
113
114
115
116
117
タスクF
pri.=0
タスクD
pri.=7
タスクC
pri.=31
タスクA
pri.=0
タスクB
pri.=99
タスクE
pri.=255
事務処理ＯＳ
リアルタイムＯＳ
110
111
122
優先順位変換
123
優先順位変換
正規化優先順
正規化優先順
120
優先順位通知
121
優先順位通知
118
ＯＳ切替え
124
優先順位比較
正規化優先順
正規化優先順
共通割込みハンドラ
ＯＳ間通信機
ＯＳコンテクスト切替
割込みマスクレベル計算
174
175
176
177

【書類名】　要約書

【要約】

【課題】

それぞれのオペレーティングシステム上で実行されるタスクの重要性を考慮して、動作させるべきオペレーティングシステムを切り替える。

【解決手段】

複数のプロセスまたはスレッドに割当てられた優先順位にしたがって実行する複数個のオペレーティングシステムはそれぞれが実行しているプロセスまたはスレッドの優先順位を通知する優先順位通知処理を有し、それぞれのオペレーティングシステムから通知された優先順位を計算機内で共通の優先順位（正規化優先順位）に変換する優先順位変換処理と、優先順位変換処理によって得られた正規化優先順位を比較して、より高い正規化優先順位を有するオペレーティングシステムを優先的に実行させる優先順位比較処理を実行する。

【選択図】　図１

認定・付加情報

特許出願の番号	平成１１年　特許願　第０４１０３２号
受付番号	５９９００１４４８８３
書類名	特許願
担当官	第七担当上席　　００９６
作成日	平成１１年　２月２３日

＜認定情報・付加情報＞

【提出日】　平成11年 2月19日

次頁無

出　願　人　履　歴　情　報

識別番号　　〔０００００５１０８〕

１．変更年月日　　１９９０年　８月３１日
［変更理由］　新規登録
住　所　　東京都千代田区神田駿河台４丁目６番地
氏　名　　株式会社日立製作所